Acta Linguistica Hungarica

AN INTERNATIONAL JOURNAL OF LINGUISTICS

Volume 41, Numbers 1–4, 1992–93

Akadémiai Kiadó, Budapest

Acta Linguist. Hung. HU ISSN 1216-8076

ACTA LINGUISTICA HUNGARICA

AN INTERNATIONAL JOURNAL OF LINGUISTICS

Acta Linguistica Hungarica publishes papers on the subjects of Finno-Ugric, Slavonic, Germanic, Oriental and Romance linguistics as well as general linguistics in English, German and French.

Acta Linguistica Hungarica is published in yearly volumes of four issues by

AKADÉMIAI KIADÓ

Publishing House of the Hungarian Academy of Sciences

H-1117 Budapest, Prielle Kornélia u. 19–35

Manuscripts should be addressed to:

ACTA LINGUISTICA HUNGARICA

H-1014 Budapest, Színház u. 5–7

Correspondence with the editors and publishers should be sent to the same address.

Orders may be placed with

AKADÉMIAI KIADÓ

H-1519 Budapest, P.O. Box 245

Acta Linguistica Hungarica is abstracted/indexed in Current Content–Arts and Humanities, Arts and Humanitites Citation Index and Bibliographie Linguistique/Linguistic Bibliography

INFORMATION FOR CONTRIBUTORS

Please provide typewritten double-spaced manuscripts with wide margins (50 characters per line, 25 lines per page). Type on one side of the paper only and number all the pages consecutively. The following information should be provided on the first page: the title of the article (or, in case of reviews, the title of the book under discussion, the name of the publisher, the place and date of publication as well as the number of pages), the author's full name and address and an abbreviated title of the paper (not exceeding 45 characters including spaces). The contribution proper should begin on page 2.

CONTENTS

PREFACE

The present collection contains papers presented at the 2nd International Conference on Computational Lexicography and Text Research (COMPLEX '92) held in Budapest, October 4 through 8, 1992, and organized jointly by the Research Institute for Linguistics of the Hungarian Academy of Sciences and the Laboratoire et d'Automatique Documentaire et Linguistique (LADL), CNRS-Université Paris 7. The aim of the conference was to bring together researchers who are experts in both computational linguistics and lexicography. Papers were invited on substantial, original, and unpublished research on all aspects of computational lexicography, including, but not limited to, the followings:

- lexical databases and electronic dictionaries,
- problems in building text corpora,
- lexical workstation,
- retrieval softwares for SGML texts,
- lexicons and other tools for machine translation.

One of the central issues addressed in this collection concerns the machine-aided compilation of dictionaries on the basis of text corpora. The computer-aided process of producing lexical descriptions is most often based on domain-specific corpora. Most lexicographic work reported in the papers is directed toward the compilation of monolingual dictionaries, though the preoccupation with the problems of bilingual dictionaries seems gaining ground. Thus, in computational lexicography one of the most intriguing questions is the use of bilingual reference corpora for translation. From among the more recent research topics mention should also be made of the analysis of compounds and idioms, of the recognition of complex patterns (verbs with their complements) and of the acquisition of semantic information. On the other hand, quite a few of the more traditional topics of computational lexicography are still not exhausted. To these belong automatic lemmatization, automatic lexical search in various types of text corpora, morphological analysis and parsing techniques.

Ferenc Kiefer

Akadémiai Kiadó, Budapest

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 5-71 (1992-93)

TOOLS FOR COMPUTER-AIDED CORPUS LEXICOGRAPHY: THE HECTOR PROJECT

BERYL T. S. ATKINS*

0. Introduction

In this paper,[1] I shall record the computational tools designed and built by members of the staff Digital Equipment Corporation in order to facilitate lexicography carried out by members of the staff of Oxford University Press [OUP]. After introducing this joint project (section 1), I shall give a brief account of the resources available to the lexicographers (section 2); next I shall describe the tools themselves (section 3); and then I shall go through in some detail the whole process of compiling a lexical entry (section 4), showing at every stage how the computer tools function, and illustrating each step with a screen dump taken in the course of the lexicographical work.

1. The Hector project

The Hector project was the name given to a collaborative venture into computer-aided lexicography undertaken by the US computer company Digital Equipment Corporation and the British publishing house Oxford University Press. It was located at Digital's Systems Research Center in Palo Alto, CA,

* B. T. S. Atkins, Lexicographical Adviser to Oxford University Press, has been a professional lexicographer for over 25 years, working on monolingual and bilingual dictionaries, and was a General Editor of the Collins-Robert English-French range. She is currently President of EURALEX (European Association for Lexicography).

[1] This is an account of a team project, managed by Mary-Claire van Leunen (for Digital Equipment Corporation) and Patrick Hanks (for Oxford University Press). The Digital team consisted of Lucille Glassman, Cynthia Hibbard, James R. Meehan, and Loretta Guarino Reid (DEC Systems Research Center, Palo Alto, CA). The principal Oxford lexicographers were Rosamund Moon and William R. Trumble, assisted at different stages by Helen Liebeck, Peter Gilliver, and Katherine Barber. My role was that of lexicographical adviser to the project, concentrating on the software specifications and including some practical lexicography. My thanks go to Mary-Claire van Leunen and Patrick Hanks for their comments on the first version of this paper.

and administered by Digital. Its purpose was twofold: to compile traditional dictionary-type lexical entries using corpus evidence, and to sense-tag the corpus lines that are being scanned in the process. It was begun in January 1991 and ended in March 1993; four members of Digital's staff worked full time on this project; several OUP lexicographers contributed to the design phase, four working in Palo Alto for a year; a number of others from both companies made contributions as required. I shall describe the project entirely from a lexicographer's point of view (for a computational view, see Glassman *et al.* (1992)), and to try to show how the computer tools made the work faster, more thorough, more consistent, and infinitely more fun.

The lexical entries constitute a source database rather than a dictionary text proper. Our purpose was to analyse the way words are used, and to record as much information as possible, which in effect meant as much information as we could manage within our own time constraints.

The manual sense-tagging of the corpus occurrences of each compiled word is not a task that has (as far as I know) been undertaken on this scale before, although lexical research has skirted round this topic for some years (see Lesk 1986; Atkins 1987; Black 1988). In the initial stages of compiling an entry it is easy to set conditions that will produce groups of occurrences to be given the same sense tag in a macro command, but as the manual tagging wears on this becomes less and less possible, and it has to be said that in the case of the words being tagged during this project (on average, those which occur between 500 and 1 000 times in the corpus) the last 50% or more of the manual sense-tagging was tedious and slow.

2. Lexicographical resources

The resources available to the lexicographers were of various types: evidence of the language in use (see **2.1**), bibliographical details of the corpus texts from which the evidence is drawn (see **2.2**), reference works about the language (see **2.3**), and the result of linguistic analysis of the English verb system (see **2.4**).

2.1. Linguistic data

The main body of evidence is to be found in the electronic corpus; this is supplemented by a file of citations collected as part of OUP's continuing reading programme.

2.1.1. The electronic text corpus

The bulk of the evidence analysed took the form of an electronic corpus of current British English, containing approximately 17.3 million words principally of written texts but also some transcribed spoken language.[2] The corpus contents were preprocessed for the lexicographers: cleaned up (punctuation checked and tagged, duplicate texts and typographical errors removed, etc.); wordclass-tagged, using a tagset of 623 tags devised for the Lund–Oslo–Bergen corpus (Johansson–Hofland 1989), grouped into 14 clusters; parsed, using the Houghton–Mifflin parser, which includes a second tagging process; and word-form occurrences and collocational frequencies computed.

2.1.2. The citations file

OUP maintains a reading programme in Oxford and (for American and Canadian texts) in New Jersey; the citations collected in the course of this programme are keyed and used by Oxford lexicographers recording new words and new usages of existing words; this continuously growing body of text was available on line to the lexicographers of the Hector Project.

2.2. Corpus catalogue

Bibliographical details of every text in the corpus could be called up by the lexicographers. A typical entry (with end tags removed) reads:

```
      <code> NiceWk
     <title> Nice Work
   <comment> novel. Booker shortlist
      <date> 1988
    <authper> David Lodge
       <age> 40–50
   <authmode> single              (= only one author)
       <sex> male
<nationality> UK
   <domicile> UK
    <compos> single               (= only one typesetter)
  <publisher> Secker and Warburg
     <place> London, UK
     <genre> written; published; books; fiction
   <samplen> 109,109              (= number of words in sample)
```

[2] This corpus was drawn from the Oxford Corpus, built along the lines described in Clear (1993); we are grateful to Jeremy Clear, Corpus Manager, OUP, for his help in building it.

2.3. Oxford reference works[3]

The following works were available on line for consultation by the lexicographers:

The Concise Oxford Dictionary [COD8], 8th edition, Oxford University Press, 1990.

The Pocket Oxford Dictionary, 8th edition, Oxford University Press, 1992.

The Oxford Dictionary of Quotations [ODQ], 3rd edition, Oxford University Press, 1980.

The Oxford Thesaurus. Compiled by L. Urdang, Oxford University Press, 1992.

The Shorter Oxford English Dictionary [SOED2] (second edition). Oxford University Press, 1993.

The attributed citations in the SOED2 and ODQ, together with those in the Citations File (see **2.1.2**), usefully complement the electronic corpus.

2.4. Verb checklist

This was adapted from Levin (1993), and consisted of a listing of verb patterns and alternations; it was available on line, and served as an aide-mémoire for lexicographers who wished to check that the corpus and citations sources offered adequate information about the constructions associated with the principal English verbs. When a lexicographer asked about a verb, the program offered one or more of the verb lists (see **4.1.4** for an example) where patterns of usage, including transitivity alternations, are stated; each pattern is exemplified, using one of the verbs to which that pattern applies, in such a way that the lexicographer can see the potential of the verb that is the subject of the query.

3. The computational tools[4]

In this paper, I wish to concentrate on the specifically lexicographical functions of the Hector tools. There are a number of more general functions which I shall

[3] We are grateful to James Howes, Head of Reference Computing, Oxford University Press, for help with the transfer to Palo Alto of dictionary data and for the OUP "Sid" dictionary browsing software which contributed to the Argus Reference Tools and the Ajax Dictionary Editor.

[4] This account owes much to the excellent documentation provided by our Digital colleagues.

not describe: idiot-proofing (for instance, they do not allow lexicographers to destroy data without being aware of it); periodic automatic saving; monitoring and recording every command that is input, together with its consequences; flexible windowing for almost all of the menu and dialogue boxes, and so on.

The lexicographers' machines were Digital 5100 workstations. The lexicographical process, including corpus searching and sense-tagging, is too complex to be carried out on a single monitor, even one with many windows (the windows are organized by the Motif Window Manager). The prototype configuration had six monitors, and these were all used and useful, but the hardware eventually decided on meant that we finished up with three (see Fig. 1). This was a relief to those whose eyesight could not cope with six screens all in different focus.

The computational tools designed to assist the lexicography fall into three main groups:

(1) those which access reference material, used in the left-hand monitor ("Atlas");

(2) those which display structured corpus texts, and perform operations on these, in the centre monitor ("Argus");

(3) those which receive, structure, record and display lexical entries and amendments to these, in the right-hand monitor ("Ajax").

This triple screen configuration allows the lexicographers to scan, sort and sense-tag the corpus data, and consult other reference sources, without losing track of the "shape" of the growing lexical entry. The cursor travels freely from one screen to another. Material may be cut from the corpus data in Argus and pasted in (as examples) into the appropriate field in Ajax.

3.1. The Atlas reference tools

The left-hand monitor ("Atlas") is the reference screen. These are the tools which present the lexicographer with processed lingustic data, offering dictionary browsing facilities, statistics on wordform occurrences and collocational frequences, and a checklist of verb patterns. In addition, the lexicographers use the Atlas monitor for requesting information about texts in the corpus, or a progress report on the project, and for checking the consistency of their compiled entries; this is also the screen used for routine tasks such as document writing and handling electronic mail.

Fig. 1
Atlas, Argus and Ajax monitors

3.1.1. Commands in Atlas

The Atlas screen is entirely flexible and is configured by the lexicographers to suit their own tastes; unlike Argus and Ajax, which each have customized screens, there are no fixed access points to any of the commands, which are simply keyed into a Unix shell window. They fall roughly into three groups, according to the type of information accessed: (a) reference work entries; (b) mainly preprocessed information such as corpus statistics; and (c) the lexicographical policy documents. The type of argument for each command is given in square brackets after it.

3.1.1.1. Accessing reference works

These display in quasi-print format the selected entry from the specified work (see **2.3**, and Fig. 10), and are as follows:

cod8 [headword]
shorter [headword]
pocket [headword]
oxthes [headword]
odq [headword]

3.1.1.2. Mainly accessing preprocessed information

beth [verb-lemma]

This command produces information about the complementation patterns and transitivity alternations in which the lemma participates (see **2.4**).

checkentry [headword]

This command checks the contents of three fields (semantic domain, register and style labels, and grammar) in a compiled entry. These may each legitimately contain only a specific set of character strings, and those which do not conform are easily identified. Later, other fields will be added to the list of those that fall within the scope of **checkentry**.

coll [wordform]

This command produces two sets of figures derived from statistical analysis of the corpus (see Church *et al.* 1990a, 1990b, 1994): mutual information (MI) and t-score. These are obtained by analysing all the words that occur in a window of five words to the right of each target word, and (separately) in a window of five words to the left, and computing for both right and left

environments those that co-occur with the keyword significantly more often than chance. The output of this command is shown on the right side of the screen in Fig. 5.

Both MI and t-score draw the lexicographer's attention to words that tend to occur together, but they have slightly different emphases. The t-test takes account of the absolute frequencies of each of the words in the corpus, so that the highest scoring items are not only strongly associated with the target word, they also tend to be very common words in their own right. MI, on the other hand, tends to pick out the more unusual items, which a lexicographer might have missed. The t-test often does better at drawing attention to the function words which co-occur with the target word; this can be particularly useful in studying syntactic points, for example verb complementation patterns. MI gives greater emphasis to content words, which is especially helpful as a tool for organizing semantic studies of the target words.

corpusdoc [text-title-abbreviation]

An abbreviation of the source document title appears alongside each corpus line. This command expands it to give fuller information from the corpus catalogue about the text it refers to (see **2.2**).

incs [lemma]

This command opens a window into the citations file (see **2.1.2**), known informally as the "incomings"; every occurrence of any form of the lemma in the citations database is offered to the lexicographer.

printentry [lemma]

This command outputs a printed version of the compiled entry in the format shown in Fig. 35.

stats [lemma]

This command calls up a summary of the number of occurrences of the word in its various forms; as well as giving the frequencies of occurrence of each lexical form, stats shows how many occurrences of the word, and of each wordform, are tagged as a noun, verb, adjective, adverb etc. The output of this command is shown on the left side of the screen in Fig. 5.

tally

This command computes the compilation to date and reports the situation, noting what entries have been compiled by which lexicographer, on what date;

their length and complexity; the percentage of the wordlist compiled and of the corpus sense-tagged at the moment of enquiry, etc.

3.1.1.3. Accessing lexicographical policy documents

The policy documents are an on-line style guide, setting out for the team of lexicographers the exact way in which various types of linguistic data must be handled during the compilation. These commands open a text window on to the policy documents file at specific points; some examples are:

comps (compounds)
gram (grammar)
ids (multiword items)
punct (punctuation)
reg (register) etc. etc.

3.2. The corpus searching and tagging tools ("Argus")

The centre monitor ("Argus") offers corpus data to the lexicographers, with a dialogue facility that allows them to specify conditions on the corpus searches.

These tools operate on raw linguistic data. At the heart of Argus is the 17.3 million-word corpus, tagged with text structuring features (e.g. those marking paragraphs, sentences and other text units, also typographical details, etc.) and grammatical features (wordclasses and some clause roles); the team tagged some of the nouns in the corpus with fairly broad semantic features such as PERSON, PLANT, VEHICLE, GARMENT, FOOD etc. These were drawn semi-automatically from an on-line thesaurus, but similar tags could of course be assigned from an on-line dictionary, using semantic taxonomies constructed from the parsed definitions, as described by Chodorow *et al.* (1985); Byrd *et al.* (1987); and Calzolari–Picchi (1988).

Argus uses these existing tags in the selection of concordance lines conforming to conditions set by the lexicographers, and displays these for sense-tagging; Argus also records the sense-tag assigned to each occurrence.

3.2.1. Commands in Argus

The command system in Argus and Ajax is very flexible. Some of the command buttons execute the command directly; others open up into a menu of further commands and options, possibly cascading to several levels; and others open a dialogue box into which the lexicographer must key some information.

The basic Argus layout is shown in Fig. 2. On the screen there are two windows, upper ("Query") and lower ("Argus"), of equal size in this illus-

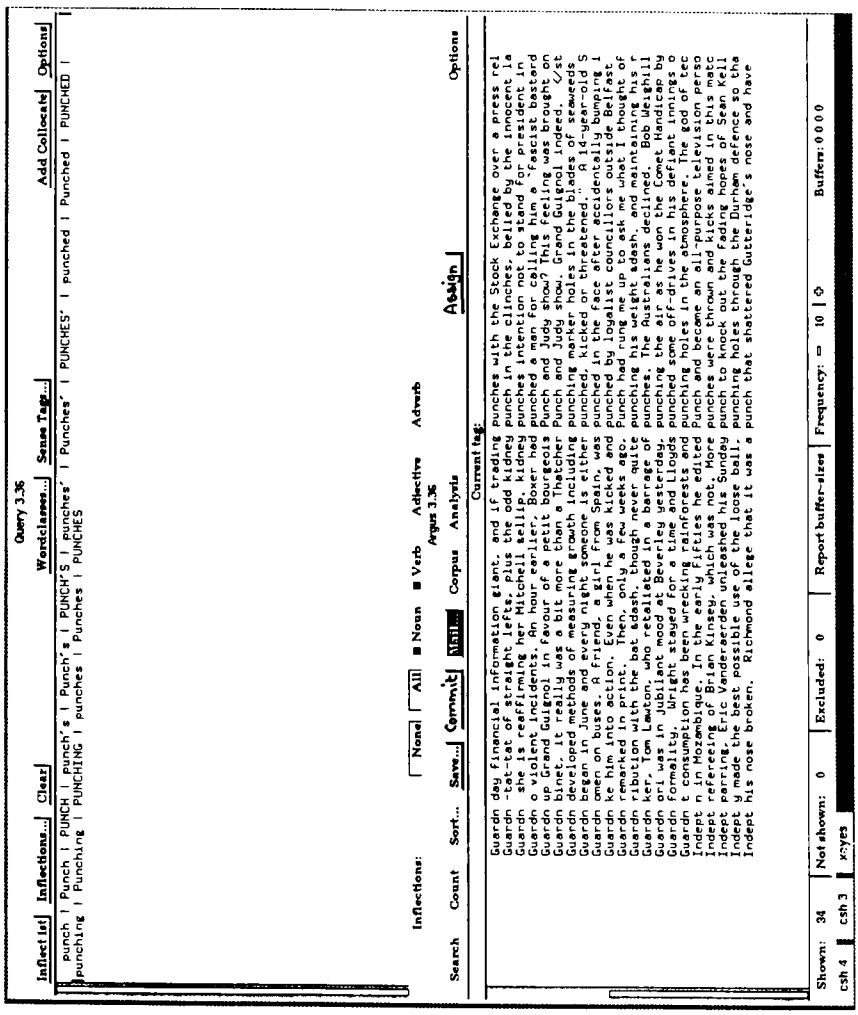


Fig. 2
Argus screen

tration, but the lexicographers can change this if they want to. Each has its own command line, which I shall describe in turn. Suspension points following some of the commands indicate the presence of a menu of options.

3.2.1.1 The "Query" window

The first three buttons in the command line (**Expand 1st, Inflections ...** and **Clear**, at the top of the screen in Fig. 2, starting from the left) all operate on the same data: the word keyed by the lexicographer into the space below. This "target" word forms the focus of corpus searches.

The word may be "expanded" into all its possible forms, covering all the typographical alternatives for all the selected morphological inflections: in the screen shown, the noun and verb forms of *punch* have been expanded. The vertical bar separating the wordforms stands for "or".

If a target word is entered without hitting "Return", the target is exactly the word form as keyed in, with no expansion. If "Return" is hit, the word is expanded typographically only (e.g. punch | Punch | PUNCH).

It is possible to key in more than one word, and hitting "Return" expands them both (or all).

Expand 1st

This command expands the first target word only, according to options selected by using the Inflections ... button.

Inflections ...

This command opens a menu of options, shown in Fig. 2, lying along the bottom of the upper "Query" window. As with all these menus, a mouse click on a button selects that option (here, "Noun" and "Verb"). The default is "All". These options are not mutually exclusive. Selecting "None" produces variant spellings, and upper and lower case alternatives, for the target word. Selecting "Adverb" for *punch* would have produced *punchly*, *punchlier* and *punchliest*, for which corpus matches are unlikely. (In the matter of corpus contents, I've come to share Napoleon's attitude to the word "impossible".)

Clear

This command clears the contents of the target word box, allowing the lexicographer to key in a new target.

Wordclass . . .

This command opens a dialogue box that allows the lexicographers to add further constraints to the search for target word occurrences, using the word-class tags in the corpus. The default is "All". The options are "Noun", "Verb", "Adjective", "Adverb" and "Other". However, the primary constraint is the output of the expansion process: the search will be performed only for word-forms appearing in the target box.

Sense Tags . . .

This command constrains the search still further, by offering the lexicographer the chance to search only for those occurrences already tagged with specific sense-tags. It opens up into a list of the active sense-tags for the target word (see **3.4**) and the lexicographer may choose them all, or some, or none. This is useful when you are looking for all the occurrences of one particular sense, or when you want to exclude, during the sense-tagging process, all the lines you have already sense-tagged. Here again, the default is "All".

Add Collocate

This command opens a new search condition area, allowing the lexicographer to key a target collocate word into the target box. This restricts the search to occurrences of the target word in the context of a specific collocate, or specific collocates.

All the options on the target word (*punch* in Fig. 2)—**Expand 1st**, **Inflections . . .**, **Clear**, **Wordclasses . . .** and **Sense Tags . . .** —are repeated here for the collocate, and in addition the dialogue box **Position** allows us to specify either a position or a range of positions for the collocate location vis-à-vis the target word. In selecting these, the default is "–5,+5" (within a range of positions from five words to the left of the target word to five words to the right). The options here are fully flexible: "+1" would restrict the collocate search to the word occurring directly after the target word, "-20,+20" would greatly extend the search window, and so on.

The **Add Collocate** command functions, if required, on the wordclass tags alone; that is, it is possible to search for all occurrences of the target word with a preposition, or noun, or any other part of speech in a certain relative position. The words which match this condition are colour-highlighted in the resultant concordances, making it easy to see patterns, find phrasal verbs etc.

There is no limit to the number of collocate boxes the lexicographer may add. The relationship between the collocates is one of conjunction ("and"); that is, if I add the two collocates *rum* and *fruit*, each in its own collocate

box, to the target word *punch*, Argus will find only concordance lines where both *rum* and *fruit* co-occur with *punch*. If I want to find all the lines where either *rum* or *fruit* co-occurs with *punch* I must enter both collocates into the same collocate box, with a vertical bar between them.

The collocate box can itself generate another collocate box (a collocate of the collocate). Positions in that box are relative to the parent collocate, not to the target word.

The facility to add collocates to both the target word and to collocates, together with the relationships of conjunction and disjunction that may be stipulated, make this search procedure extremely flexible (if rather complex and time-consuming).

Options

This command allows the lexicographer to reposition the Query window, and opens the following menu:

Move to Screen 0
Move to Screen 1
Move to Screen 2.

3.2.1.2. The "Argus" window

This window displays the result of a search and most of the commands relate to the concordance lines in the window. The command line with buttons Search, Count etc. lies at the top of the window (horizontally, centre screen, in Fig. 2); the six boxes (Shown, Not shown etc.) at the foot of the window report on the search. I shall first describe the command buttons, starting from the left, and after them the search results boxes.

Search

This button starts a search, using input from the Query window. The resultant concordance lines start to scroll in the display window. The context of the target word is approximately 50 characters to the right and 50 to the left; an abbreviation identifying the source text lies on the left of the line (in Fig. 2, these indicate the British newspapers *The Guardian* and *The Independent*), while words matching the collocate specifications are colour-highlighted. The blank area to the left of the lines is where the lexicographers insert sense tags.

The lines appear in the order that the texts are accessed in the corpus. After a few weeks of experimentation, we decided that initially it would be most useful for the lexicographers to have the written texts sorted first on

genre (with newspapers first, and fiction last), and within this primary sort, resorted on title, alphabetically; the transcribed spoken texts come last.

Count

This command simply counts the occurrences that match the search conditions, without displaying the concordances, and displays the total in the **Not Shown** box below the concordance window. It is much faster than **Search.**

Sort

This command takes the output of the **Search** command and sorts it according to several options. There are two sorts, a primary and a secondary (see Fig. 7). Each offers the same options, viz:

Words At Right	=	alphabetically sorted on right context
Words At Left	=	alphabetically sorted on left context
Wordform	=	sorted on the wordform of the target word
Corpus	=	sorted in order of genre in the corpus
Sense	=	sorted on the sense tags that have been input

The default here is "Corpus" in both cases. Sorting on right and left context does not operate beyond the sentence boundary.

Save

This command puts the occurrences that result from a Search into a file according to further specification from the lexicographer. The options here are either to make a file of KWIC concordances as displayed on the screen, or to make a file of the full sentences in which these occurrences figure.

Commit

This command (often invisible in the screen dumps because of the colours on the monitor) saves the sense tags that the lexicographer has assigned to the corpus occurrences.

Mail

This command opens an instant hot line to the lexicographers' "minders", the Digital computer scientists who keep the software running and continue to add enrich it as the lexicography develops. Mail is a streamlined bug-reporting system.

Corpus

This command, called when the cursor is on a concordance line in the Argus display window, opens a window into the corpus. The options here are "Partial" (a window of 1 000 characters to the left and right of the target word) or "Entire" (the whole corpus).

Analysis

This command, called when the cursor is on a concordance line, opens a scrollable window where are displayed the wordclass tags and clause analysis data, including subject and predicate marking, that the sentence carries.

Assign

Here again, the label on the command button does not come out well in the screen dumps. This command assigns a designated sense tag (the one appearing at Current tag just below it) to a previously selected line or batch of lines in the display window, thus allowing for batch tagging.

Options

This command opens up a menu of options. In addition to those for moving the Argus window to another screen, it includes several of direct interest in debugging operations, and the option to Quit, i.e. close down Argus in an orderly way.

Shown

is the first of the search results boxes lying along the base of the Argus window. In it is displayed the count of occurrences that match the search conditions, and for which KWIC concordances are being displayed.

Not shown

This command also gives the total of matches, this time for the output of the Count command, when the matches are displayed on the screen.

The other boxes are concerned with the running of the software, and normally used only by the computer scientists in the team.

3.3. The dictionary entry editor ("Ajax")

The right-hand monitor ("Ajax") houses the entry-building software; it is there that the lexicographers call up skeleton template entries and flesh them out into full entries.

These tools provide a structured environment in which the lexicographers compile lexical entries. This can be done from scratch, starting from a blank template, or a COD8 entry can be called up in Ajax for adaptation for the new dictionary-database in preparation.

The structure of the entries in Ajax is based on deconstructions (or parses) of the COD8 entries. Ten types of entry structure were identified, and the parses drawn up.[5]

The types of entries for which parses were made are:

abbreviation_entry
affix_entry
biographical_entry
contraction_entry
crossref_entry
geographical_entry
given_name_entry
institution_entry
lexical_entry
nonassimilated_entry

The least complicated of these (that for a foreign loan word) is given here, as an example:

```
nonassimilated_entry  uid=###.###.###.###
                      hidden=FALSE
                      ord=numlet(0)
                      sort=()

    free_text
    pronunciation
    variant_spelling*
    decoration*
    see_also_xr*
    source
    na_typology
    wordclass_sequence+
    derivative_sequence*
    etymology
    note*
```

[5] This is the work of Rosamund Moon.

where

X+ = one or more X
X* = zero or more X

The parse (like all such) formalizes the structure of the entry-type it describes and defines. Here is one "nonassimilated" entry in print:

bonhomie /ˌbɒnɒ'mi:/ *n.* geniality: good-natured friendliness. [F.f. *bonhomme* good fellow]

Fig. 3
COD8 "non-assimilated" entry

The relationship between the lines of the parse and the elements of the entry is transparent enough.

This work informed the whole design of Ajax, and of course also allows Ajax to validate input data in certain fields, check that no essential part of an entry has been omitted, and order the entry components.

3.3.1. Commands in Ajax

The Ajax screen shown in Fig. 4 is known informally as "proto-Ajax". It is not a template for a full lexical entry, simply an outline format to be used for the basic structuring and compiling stages. We started off with a full dictionary entry template ("Ajax" proper), but this was so complex that, for all but the very short entries, no amount of manipulation would allow enough material on the screen at any given moment. We then reduced the number of fields and options, for the working draft template ("proto-Ajax"), and devised the "icon button" method of accessing other fields (see **3.3.1.2**).

The design of "proto-Ajax", and indeed of full Ajax, developed with the project. As the lexicography continued, and the computer scientists responded to requests from the lexicographers, the design was modified, and the operations got slicker and the screen more adapted to dictionary compiling. Proto-Ajax gives the lexicographers the ability to write a considerable amount of detail about each dictionary sense, and at the click of a mouse either to increase the scope of the entry by adding another data field, or to close up the data fields and summarize on the screen a lot of senses at the same time, while even more of the entry is visible in quasi-dictionary format through the

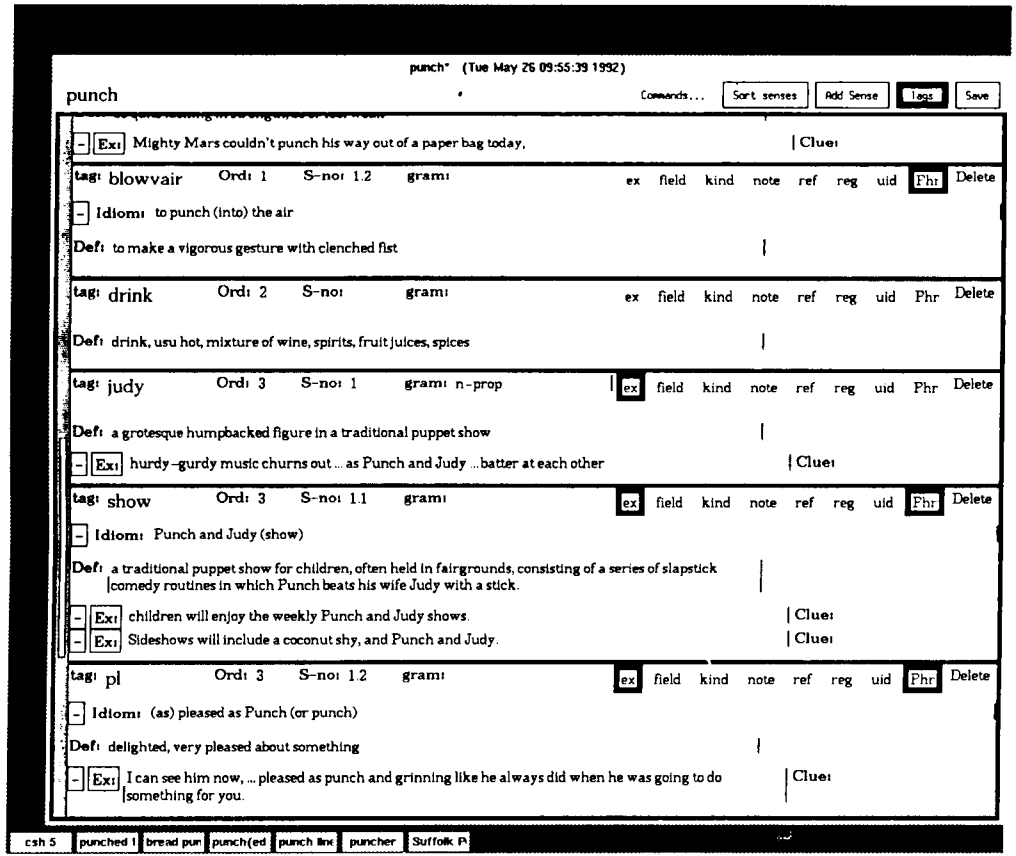


Fig. 4
Ajax screen

Show command (see **3.3.1.1**). Throughout this paper, "Ajax" refers to the "proto-Ajax" that was used to construct the demonstration entry.

3.3.1.1. The Ajax command line

In Fig. 4 the headword (*punch*) is entered in the extreme left corner of the command line at the top of the screen, where the headword normally appears in print dictionaries. Keying in a different word here automatically closes the current entry (with safeguards) and calls the entry for the new word, or if there is none, a blank template.

Most of the commands are accessed from a menu under the **Commands...**" box; the four most frequently used (**Sort senses**, **Add Sense**, **Tags** and **Save**) are buttons in their own right.

The command line contains the following options:

Commands...

This opens up a menu of commands:

Another Ajax
Odel Entry
Cod8 Entry
Full Ajax
Copy
Show
Alphabetically
Quit
Move to Screen 0
Move to Screen 1
Move to Screen 2

Another Ajax

This command brings up another blank template and holds both templates in the active mode, so that mnemonic tags from both entries are "official" at the same time in Argus (useful if you are writing entries for, say, *punch* and *punch up* at the same time, since Argus will not distinguish these occurrences in the corpus search).

Odel Entry

This command ("Odel" is the name of the dictionary-database being compiled) calls up a compiled entry and displays it in Ajax format.

COD8 Entry

This command does the same for a COD8 entry (useful if it only needs tweaking to fit the new dictionary).

Copy

This command copies the current Ajax entry to a file.

Show

This command displays the current entry in something like print dictionary format. This is an excellent tool and one that is in constant use. It is very difficult to compile on-line without being able to take an overview of the entry very frequently. **Show** gives us the facility to do this.

Alphabetically

This command opens a dialogue box and invites the insertion of a keyword that will serve to alphabetize the entry within the headword list, if the form of the headword makes it impossible to do this automatically (e.g. to put "42nd Street" amongst the Fs).

Quit

This command exits Ajax, with appropriate precautions.

Move to Screen 0 / 1 / 2

The last three commands in the menu allow repositioning of the Ajax material (few lexicographers use this facility).

There are three other top-level commands in Ajax:

Sort Senses

This command (the next button in the Ajax command line, see Fig. 4) sorts the senses according to the numbers entered at **Ord** (= ordinal number) and **S-no**: (= sense number) in each sense frame, and displays the newly sorted version on the screen. It is not always necessary to be able to see the very latest ordering displayed in the Ajax screen, and the lexicographers, unwilling to add even a few seconds to the time spent waiting, rejected automatic resorting of the entry each time a sense number was changed.

Add Sense

This command adds another sense frame at the bottom of the Ajax template. Each frame holds the contents of a dictionary sense. It is possible to add an empty frame at a particular point within the entry by positioning the cursor and striking an initialized function key.

Tags

This command transmits the mnemonic tags to Argus (the corpus search tool), at which point the tags become official, each with a unique identity in the database. Argus responds by expanding the **Sense Tags** button in the top line of the Query window (this may be seen in Fig. 12) and listing the officially recognized mnemonics that will be accepted as sense tags against corpus lines. Any mnemonic tag in Ajax can of course be eliminated, or changed. This is done by making the changes in Ajax, and toggling **Tags** off, and on again; Argus responds by listing the new set of tags. Block de-assignment and reassignment of tags to corpus lines is also possible.

Save

The **Save** button saves all the Ajax data inserted since the last save.

3.3.1.2. Buttons in the Ajax sense frames

I shall explain here the options represented by these buttons; their use will become apparent as their role is described in sections **4.2** and **4.3**.

tag

Beside this label is a text field into which the lexicographer keys the mnemonic chosen for that particular sense (in Fig. 4, for instance, 'blowvair', 'drink', 'judy' etc.).

Ord and S-no

These also label text fields, which hold numbers assigned to homograph headwords (**Ord**) and sense divisions within the entry (**S-no**).

gram

This text field receives the abbreviations for parts of speech and other grammatical information to appear in the dictionary.

The other (icon) buttons all operate in the same way: one mouse click repositions the button against the left margin of the frame at the correct point in the entry vis-à-vis the other data in the entry (for instance, the various ex

and **Idiom** buttons in Fig. 4), and opens a text field for lexicographers to key data into, together with a "minus" button which will kill the field, if they change their mind later. If a second text field of the same type (for instance, **ex** or **field** or **note** or **ref**) is needed within one sense frame, this can be called up by clicking on the first. Another mouse click on the **ex** or **field** (etc.) icon at the top of the frame closes any active fields of that type; if there are more than one, it iconifies them together in one button.

The data types, which correspond to items in traditional dictionary entries, or to information to be held in a database although perhaps not to appear in a print dictionary, are as follows:

ex

dictionary example

field

semantic field or domain marker

kind

a shorthand name for a field that holds listings of compounds of which this headword is an element (e.g. in the Ajax entry for *punch*, in the frame for the "blow" sense, the **kind** field might hold *kidney punch* and *rabbit punch*); the contents of this field will not necessarily appear in the dictionary

note

from lexicographer to lexicographer, not for publication

ref

cross-reference to another entry

reg

register, style etc. marker

uid

displays the unique identity number of that sense in the database

Phr

allows the insertion of a multiword lexical item of which the headword is one element (idioms and phrasal verbs are identified separately within this area)

Delete

deletes the whole sense frame instantly

3.4. Ajax–Argus intercommunication: sense-tagging

The link between Ajax and Argus is the system of sense-tagging. During the compiling process, the lexicographer assigns to each occurrence of the headword in the corpus a tag relating it to one of the senses of the lexical entry. The words for which entries are being compiled during this phase of the project have on average between 500 and 1 000 occurrences in the corpus. A method had to be found of tagging each occurrence on its first (and possibly only) appearance on the Argus screen—there was no question of sense-tagging the corpus after the entry was fully compiled, as that would have doubled the lexicographical work. At the same time, the lexicographers had to be free to change their minds as often as they needed to, at any point in the compiling, over the number and order of the dictionary senses of the target headword. And they had to be able to do that without needing to re-tag the corpus occurrences that had already been sense-tagged.

This is the system that was devised: when lexicographers begin to draft an entry on the Ajax screen, they give each sense a mnemonic tag; for instance, one sense of *punch* might be tagged 'drink', one 'judy' and one 'show', as in Fig. 4. To each tag Ajax assigns a unique identity number (incorporating the headword in it), and will therefore not accept two identical mnemonics within the same headword entry. The compiler hits a key to transmit these tags to Argus. Beside each KWIC concordance line on the Argus screen is a "sense-tag box" (in Fig. 2, this lies in the empty column to the left of the KWIC concordance lines); Argus will accept in that box only "official" tags, i.e. mnemonics which have been duly transmitted from Ajax (such tags are visible in Fig. 23). When two senses are merged in the lexical entry (in Ajax), the concordance lines may be retagged in a block in Argus. When senses are moved around in the Ajax lexical entry, this has no effect on the corpus sense-tags in Argus, as the unique identity numbers remain the same. When a tag in the entry is renamed with a different mnemonic, this is transmitted to Argus, and the corpus mnemonics are duly changed.

I shall now describe how, with these tools at my disposal, I drafted an entry for the word *punch*, tagging its senses into the corpus, and I shall try to show how the various Hector tools function at each point in the compilation.

4. Entry compiling and sense tagging

The work of a lexicographer is essentially one of analysis (discovering, understanding, structuring and recording the facts about the word to be described) and synthesis (from this ordered set of facts, selecting those appropriate to the dictionary-database being compiled, and setting them out in the way most helpful to the typical user of the eventual dictionary). The lexicographical work goes into a database recording the grammar, meaning, and use of words in current English, which will in due course constitute the core of a major new Oxford dictionary.

There is no canonical method of compiling dictionary entries from corpus data. All of us do it differently. But although techniques vary, the essential operations are the same for everyone. This diversity proved an enriching and challenging factor in the Hector project, since each lexicographer had something new to contribute to the demands made on the software. Perhaps the greatest challenge of course is the requirement that every operation should be carried out with maximum speed.

4.1. Getting a fix on the word

The first step is to get acquainted with the word: to see how it behaves, to begin to distinguish large chunks of meaning in its semantics, to see what wordclasses it belongs to, to start to notice what are its commonest forms, to learn its collocation patterns. In the Hector project, getting a fix on the word before starting to compile an entry involves studying the corpus statistics (on the Atlas screen), scanning the corpus (in Argus), and consulting other dictionaries (in Atlas).

4.1.1. Corpus statistics

The two commands available are described in **3.1.1.2.** The command "**stats punch**" calls up a summary of the occurrences of *punch* in its various lexical and morphological forms (on the left half of the screen in Fig. 5). It could be useful to note that the word occurs so often with a capital P (no doubt as the name of the British humorous magazine, now alas defunct, and in the expression "Punch and Judy"), because when it comes to searching for blocks of same-sense citations in the concordances, this form should prove easy to find, and pull out a rich haul of occurrences to be sense-tagged in a block.

The command "coll punch" (on the right half of the screen) produces an extract from the collocational frequencies file. This command operates on wordforms only (hence the disparity in the frequency of occurrence between

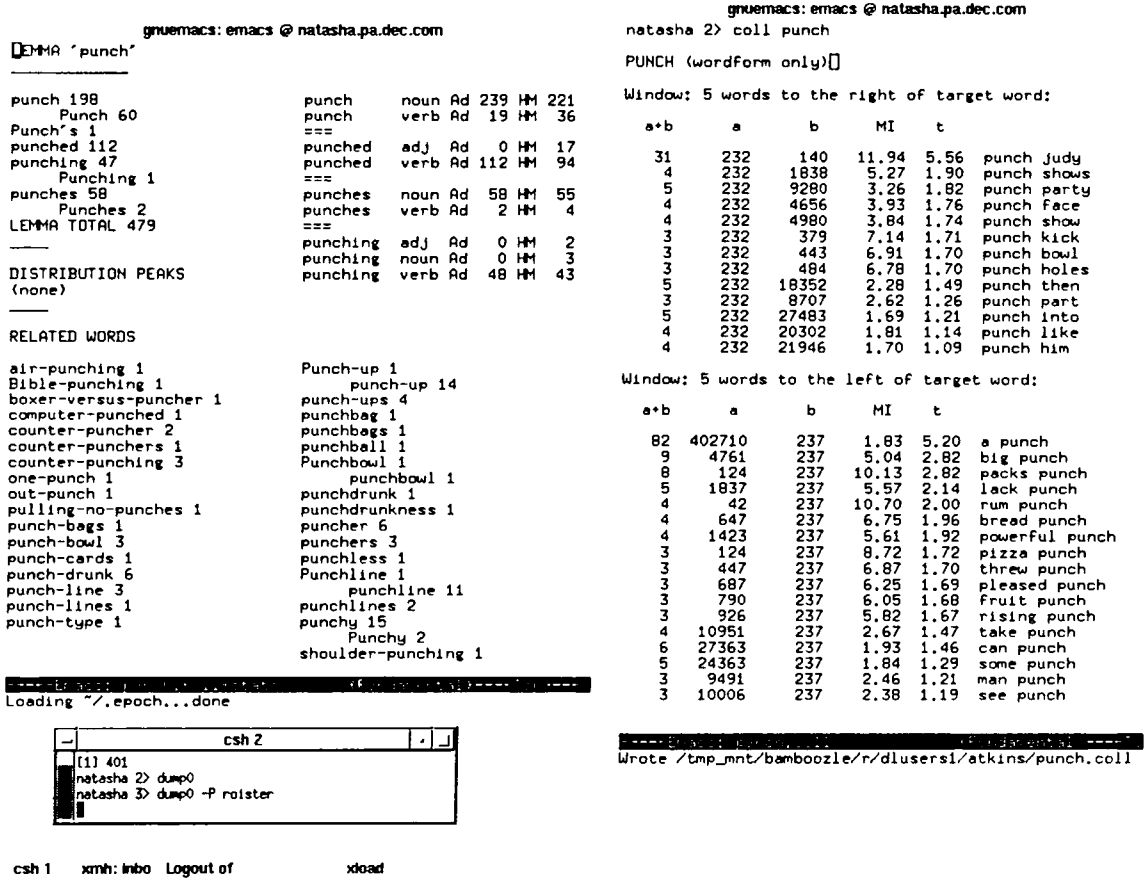


Fig. 5
Response to "stats punch" (left) and "coll punch" right

the output of **coll** and that of **stats**). **Coll** looks at collocational frequencies of the wordform 'punch' not the lemma *punch*.

In Fig. 5, the collocates of 'punch' are listed in t-score order. In this case the t-test picks out as most significant among the left collocates the cooccurrence of *a* and *punch* (a point with some syntactic significance). By contrast, the highest scoring items by the MI test are *rum* with *punch* and *packs* with *punch*, both drawing the lexicographer's attention to important multiword items: the compound "rum punch" and the idiom "to pack a (powerful etc.) punch". Given the difference between the two tests, it is all the more remarkable that both of them agree that 'judy' is far and away the most significant right collocate of the wordform 'punch'. Clearly, the Punch and Judy show continues to play a major role in English culture.

4.1.2. Scanning the corpus

The first step is to tell Argus the target word—in this case, *punch*. This word is keyed in to the target box in the Query screen (see Fig. 6), and the **Inflections** key is hit to bring up the Inflections options (central on the screen). Since I want to scan through the range of corpus lines, I hit the "Noun" and "Verb" buttons, identifying these wordclasses as active for the expansion process. When I hit the **Inflect 1st** button, Argus expands the target word as a noun and as a verb, and the various lexical forms appear (as in Fig. 6), separated by "or" bars. These forms are the target for the Argus searches, and will remain so until I change them.

There is now the option of simply counting the occurrences that match the Search condition, by hitting the **Count** button in the command line of the lower Argus screen; the result of the count is given in the **Not Shown** box, but no matches are displayed. This is much quicker than pulling up concordance lines from the corpus, but in this instance I know what the count would be (the **stats** command has already reported 479 occurrences of the lemma), so I hit the **Search** button and the KWIC concordances for *punch* start to scroll.

At this point, unconstrained by any Search conditions, the occurrences appear in the order that the texts are accessed in the corpus. As the corpus lines scroll past, I start to notice points about the context of *punch* in its various uses. I can stop the Search scrolling when I am ready to move to the next operation, or I can leave it until all instances of *punch* are collected. In either case, I can scroll up and down through the concordance lines that the Search has produced.

Normally, at this point I wish to start structuring the data a little. I hit the **Sort** key on the Argus command line, and the options are displayed.

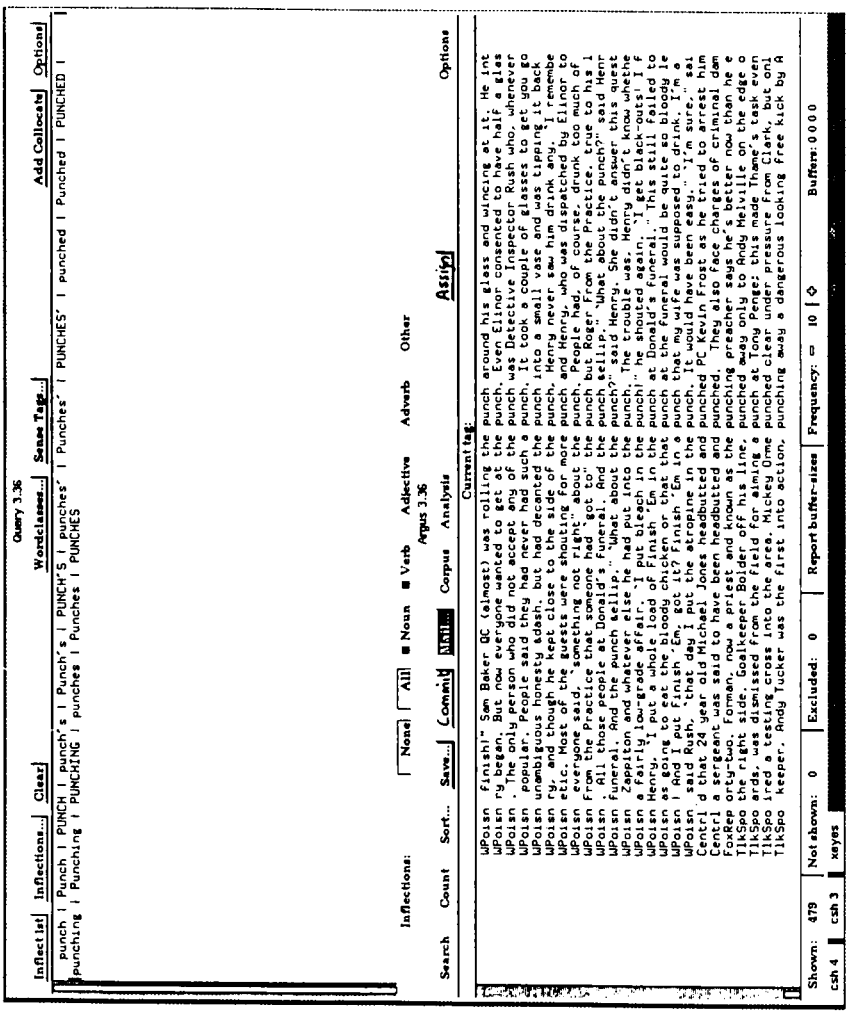


Fig. 6
(Argus) *punch* expanded, with KWIC concordances, unsorted

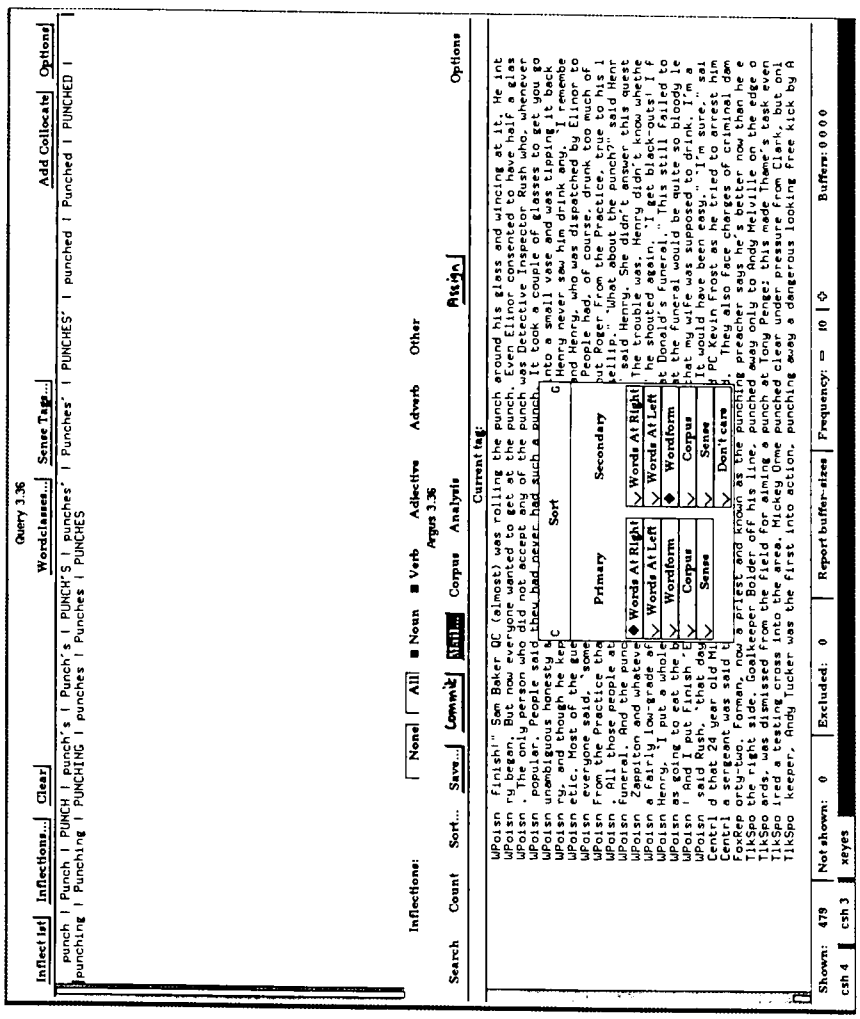


Fig. 7
(Argus) showing Sort options

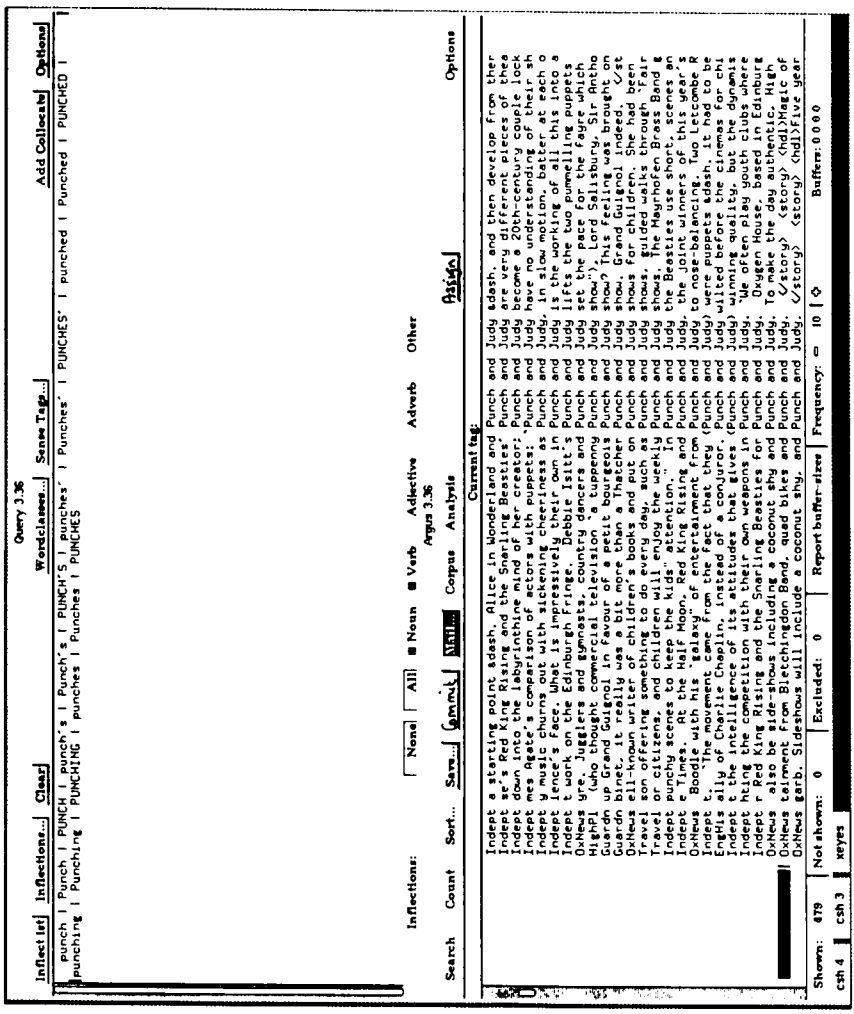


Fig. 8

(Argus) concordances sorted on right context

I choose to have the lines sorted first on the basis of right context ("Words At Right"), and within that primary sort on wordform (see Fig. 7). Experience suggests that for a word like *punch*, this is the most useful first sort.

The sorted lines appear almost instantaneously (see Fig. 8), and—still trying to get the "feel" of the word—I scroll through them. The right context sort brings out very clearly some of the patterning highlighted by the statistics output from the coll command (see **4.1.1**). "Punch and Judy" is clearly a common collocation.

Further down the lines alphabetically sorted on right context we come to the pronoun objects (*him* and *her*) of the verb *punch* (see Fig. 9). People are punched on the nose, in the face, in the stomach, in the mouth—indeed, in most body parts. This common patterning must show up in the finished entry.

Other patterns are noted, as the scrolling continues, and gradually the word starts to become familiar, its profile begins to emerge from the mists of the language, distinctive features are discerned, and I am nearly ready to make a start on the first draft of the entry.

4.1.3. Other dictionaries

Before I do that, however, I want to look at other lexicographers' views of this word. The dictionary closest to the one I am writing is COD8, so I move to the Atlas screen and call up the entry for *punch* in that dictionary (much faster, of course, than leafing through the book). Figure 10 shows the response in the Atlas screen to the commands **cod8** and **shorter**.

4.1.4. Verb patterns

Lastly, to remind myself of the potential of the verb, I key the command "beth punch", in the Atlas monitor. I am offered excerpts from sections **8.2** and **9.1** of the verb listings, showing the various patternings that apply to verbs of similar semantic content to *punch*. In the extract below, the verbs *cut*, *swat* and *punch* itself are used to exemplify these patterns:

*** (+ hole)

to	VERB	a hole in <B>	
to	cut	a hole in sth	

*** (+ resultative)

to	VERB	<B>	resultative
to	cut	sth	open

*** (+ bodypart)

to	VERB	<C>'s	bodypart
to	punch	sb's	nose

*** (bodypart possessor ascension)

to	VERB	<C>	on bodypart
to	punch	sb	on the nose

*** (conative alternation)

to	VERB	at	<B>
to	swat	at	sth

*** (zero nominal)

to	give <B>	a VERB-NOUN	
to	give sb	a punch	
to	get	a VERB-NOUN	on bodypart
to	get	a punch	on the nose

4.2. The first outline entry and sense-tagging the corpus

A first sketch is made in Ajax, where a few broad senses are identified and the mnemonics assigned. It is clear, both from the first scan of the corpus, and from the entries in other dictionaries, that there are certain distinct senses to be identified for the noun and the verb *punch*.

I start by setting up mnemonic tags for these senses, both as noun and as verb, in the Ajax template on the right-hand screen (see Fig. 11 where the mnemonics appear in large bold type against tag on the left of the screen). Among these initial senses are: striking a blow with clenched fist (verb and noun, with mnemonics 'blowv' and 'blown' respectively); making a hole in something ('pierce'); the tool that does this ('tool'); vigour and effectiveness ('vigr'); the fruit drink ('drink'); and the puppet ('judy').

The design of this database calls for a separation of noun and verb senses; wordclass tags in the corpus allow a search for noun occurrences and verb occurrences separately. It is therefore practical to group the verb senses together and the nouns together from the start of the lexicography. I try to do this when establishing my first sense mnemonics.

Figure 11 shows the first few senses sketched out. It is not necessary to enter Ajax tags in any particular order, since it is always possible to insert a new sense frame at any point in the outline entry, or to kill one by hitting the **Delete** key within the sense frame. Once the sense numbers (**S-no**: in Fig. 11)

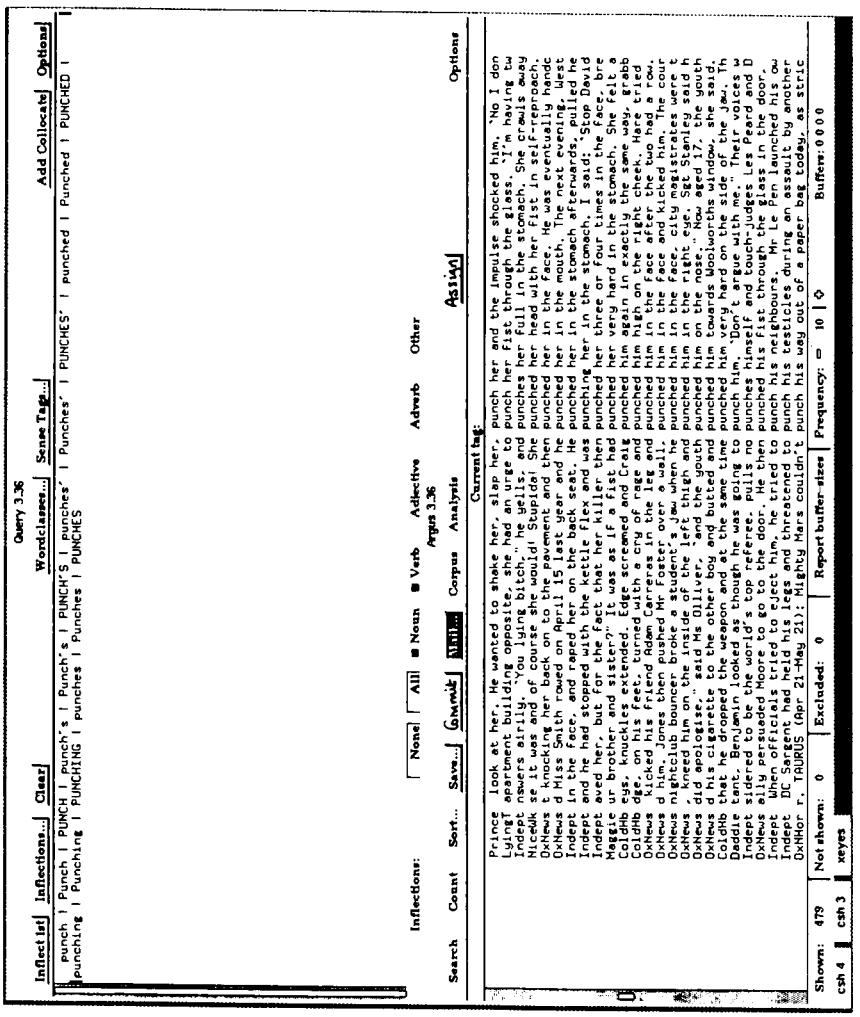


Fig. 9
(Argus) typical complements of verb *punch*

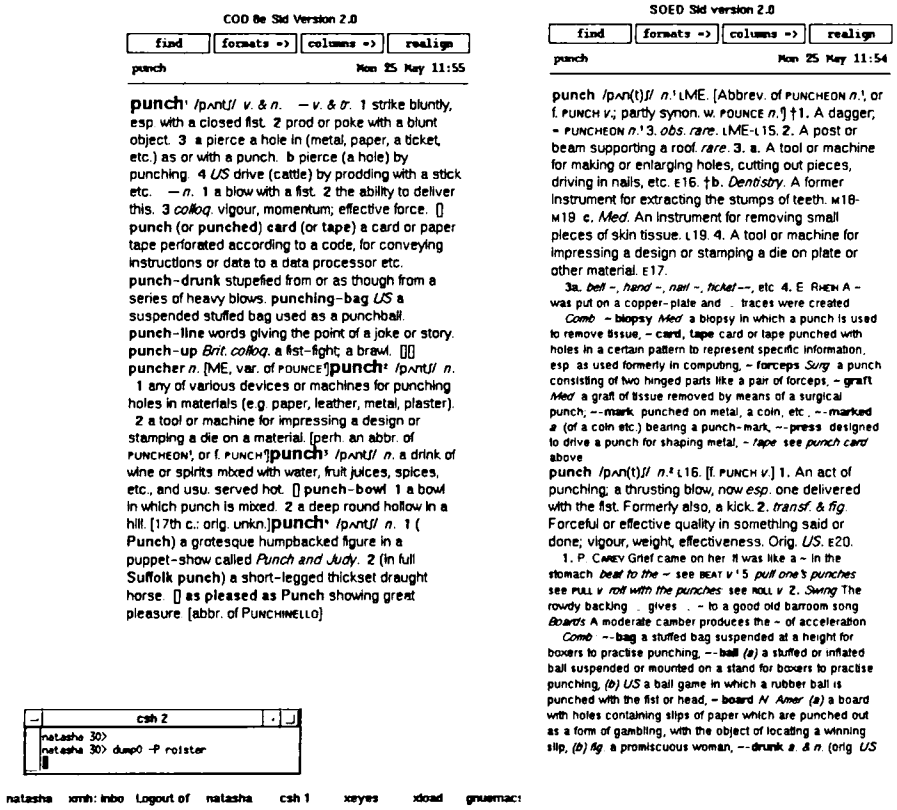


Fig. 10
Response to "cod8 punch" (left) and "shorter punch" (right)

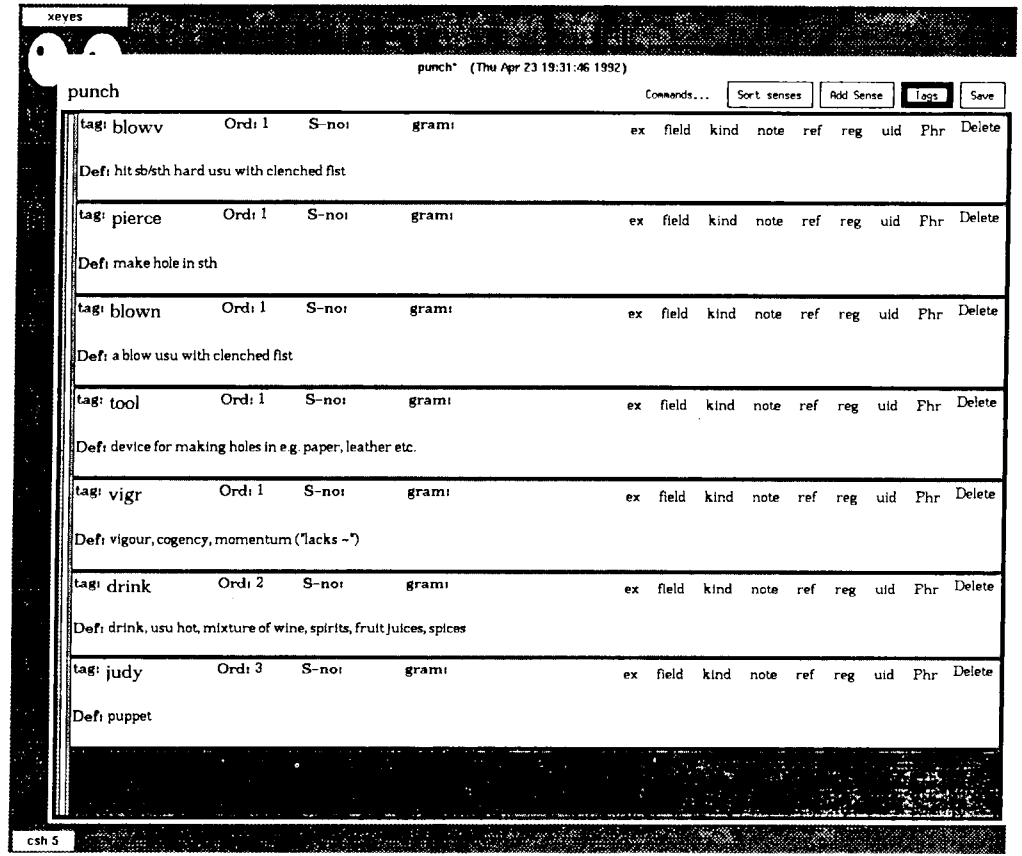


Fig. 11
(Ajax) start-up template with the first few *punch* mnemonics

are entered, Ajax knows the ordering of the draft entry, and will reorder the sense frames on the screen when the **Sort senses** command button is hit.

When **Tags** is hit in Ajax, the mnemonics become official in Argus. On the centre screen (see Fig. 12), Argus displays the sense tags available for use, and I start working through the occurrences of *punch* in the corpus, gradually building the draft dictionary entry, and sense-tagging the 479 corpus lines. The objective, of course, is to tag them as far as possible in batches, and the tools are designed to facilitate this.

First, I decide to look only at noun occurrences. This involves resetting the target of Argus's search procedure, by hitting the **Clear** button in the top command line of the Query window (see Fig. 9), which empties the text area. After keying in 'punch' again, and altering the Inflections options to read "Noun" only, a mouse click on **Inflect 1st** expands the wordform 'punch' as a noun (see Fig. 13). I also put a condition on the search by hitting the **Wordclasses** button, which opens a further band of options Wordclasses at the foot of the Query window, and opting for "Noun". This tells Argus to read the wordclass tags in the corpus and select only occurrences of *punch* tagged as a noun (two taggers have pre-tagged the corpus, neither wholly reliably, and at this point Argus offers any word that either of the taggers has identified as a noun; some verbs creep in under the net). I hit **Search** in the Argus window and the corpus lines start to scroll (see Fig. 13). I sort them on the right context, and on wordform.

It immediately becomes apparent—from phrases such as "editor of Punch"—that I need a mnemonic for *Punch* as the name of the magazine, and I add that ('mag') to the Ajax template. When I toggle the Ajax Tags button, Argus accepts the mnemonic and adds it to the displayed list.

On scrolling through the nouns, I realize that a search for "Punch and Judy" will bring up a batch to be sense-tagged at a stroke. This is done by adding a collocate to the search constraints.

A mouse click on the **Add Collocate** button in the top command line of the Argus window opens a new search condition area; this may be seen below in Fig. 14. I key in 'Judy' (which needs no expanding) and change the **Position** default setting to +2. Argus should display all concordances where the word 'Judy' appears as the second word after the target word. A mouse click on **Search** produces 28 lines that match the search conditions (*Punch* tagged as a noun, with *Judy* in position +2). At this point, the lines being correctly sense-tagged, I want to save these tags into the corpus text. To do this, I hit the Commit button in the command line of the lower half of the Argus screen. Tags which have been committed can be changed later, if need be.

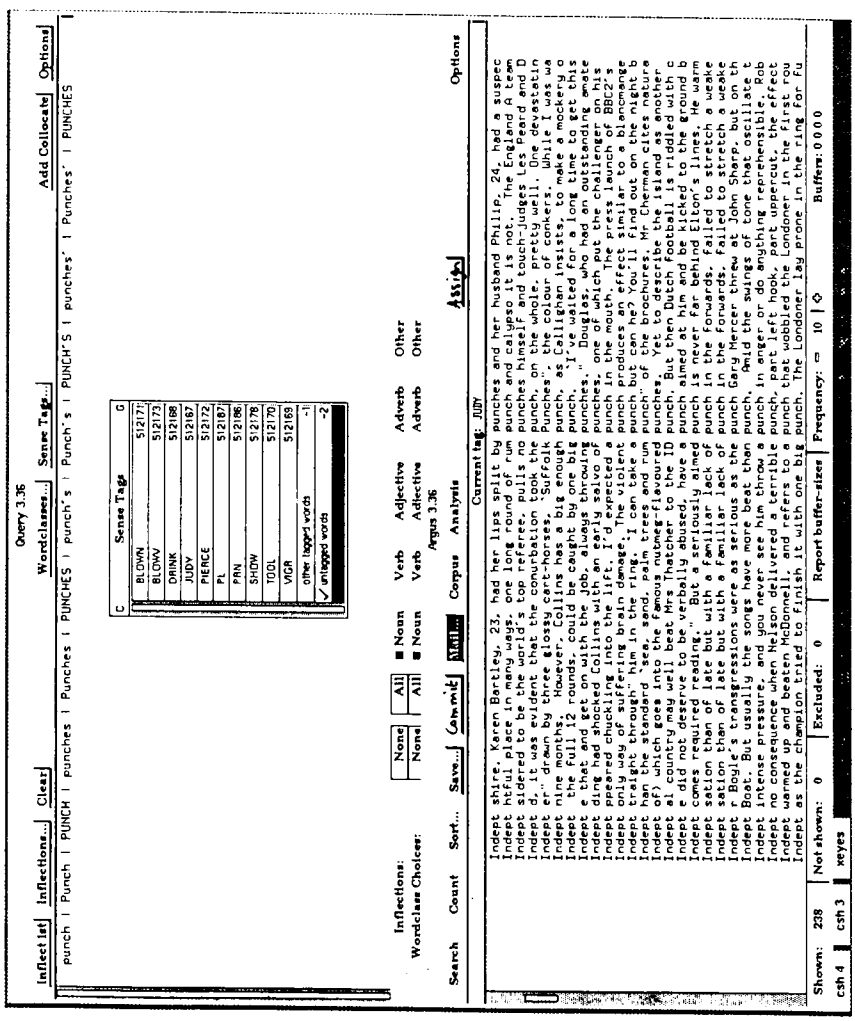


Fig. 12
(Argus) showing "official" sence tags received from Ajax

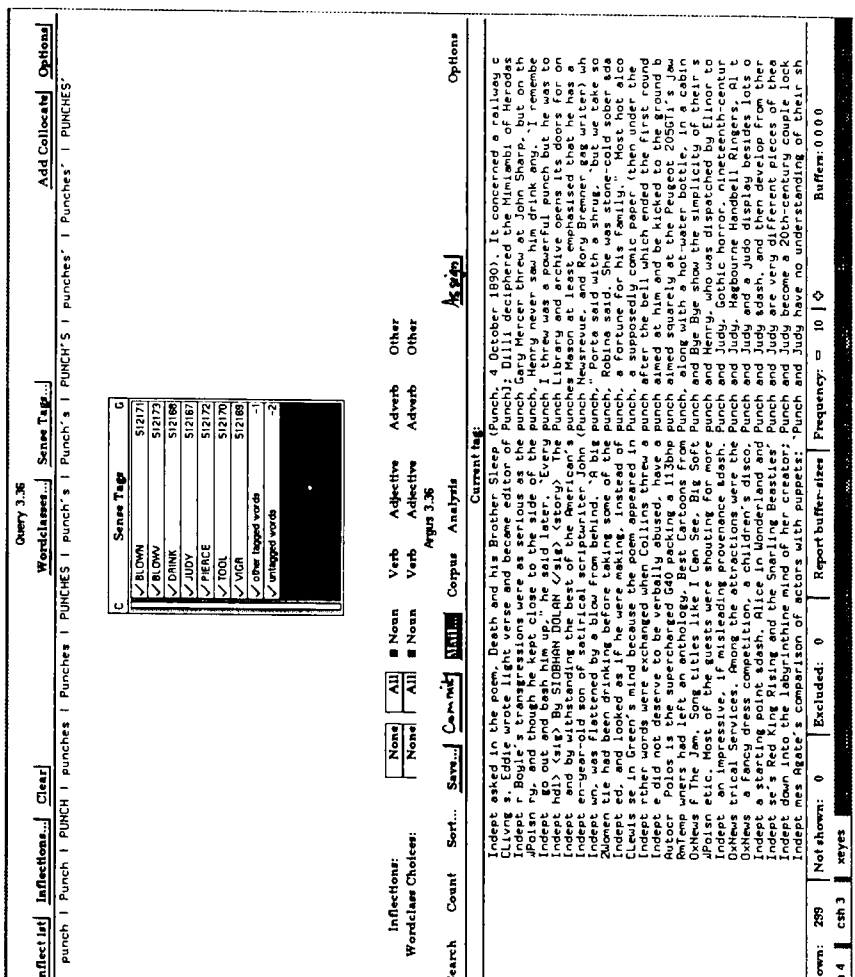


Fig. 13
(Argus) noun *punch* lines sorted on right context and wordform

I note the phrase "Punch and Judy show", which occurs six times, and want to hold it as a phrase within the draft entry. I adapt the Ajax draft entry accordingly, by inserting at the appropriate point of the entry another sense frame to hold it; by giving it the mnemonic "show" and clicking on **Tags** to make that official; and by clicking on a **Phr** button and keying in "Punch and Judy show" in the text field that appears. I also add a working definition, and an **ex** (example) field. To complete this tiny part of the entry, I return to the corpus lines in Argus to find a good example. Reading these more carefully, I realize that "Punch and Judy show" is often abbreviated to "Punch and Judy" (e.g. "Sideshows will include a coconut shy, and Punch and Judy"), and I adapt the entry accordingly. These sections are partly visible in Ajax format in Fig. 4, and a later version of them appears as 3:1.1 and 3:1.2 in Fig. 17.

I then return to the Argus window, to sense-tag the "Punch and Judy" lines. Those in which 'Punch' and 'Judy' refer to individual characters or puppets will receive the 'judy' tag, while those referring to the puppet show will get tagged as 'show'. To do this as economically as possible, I make 'show' the current tag, by keying it into the **Current tag**: field above the corpus lines. After that, the tag may be assigned to designated lines by simply blocking them out and hitting the **Assign** button. In this way, all the "Punch and Judy show" lines are tagged with the mnemonic 'show'. When I am sure that the tagging is correct, I hit the **Commit** button in the Argus command line and these sense tags are written into the corpus.

Now it is time to adapt the dictionary draft in Ajax by bracketing the word "show", since it is an optional element in the phrase, and I decide to hold an example of each form for the time being. To gather corpus examples, I scroll down through the Argus lines until I find a likely-looking citation—"children will enjoy the weekly Punch and Judy shows"; I block this out in the Argus screen, move the cursor to the correct spot in Ajax, and paste it in to the dictionary entry as an example. I do the same again with an example of "Punch and Judy" in the sense of "Punch and Judy show", noting for future reference the curious syntax of the line selected: "Sideshows will include a coconut shy, and Punch and Judy".

Back in Argus, I block out all the corpus lines visible on the screen, change the **Current tag** to 'judy' and assign the tag with one mouse click to all the remaining "Punch and Judy" lines (those already tagged with 'show' are not changed). I "commit" these tags to the corpus, and look for an example of *Punch* used to designate the puppet, or the character. The line "as Punch and Judy, in slow motion, batter at each . . . " catches my eye, but is incomplete.

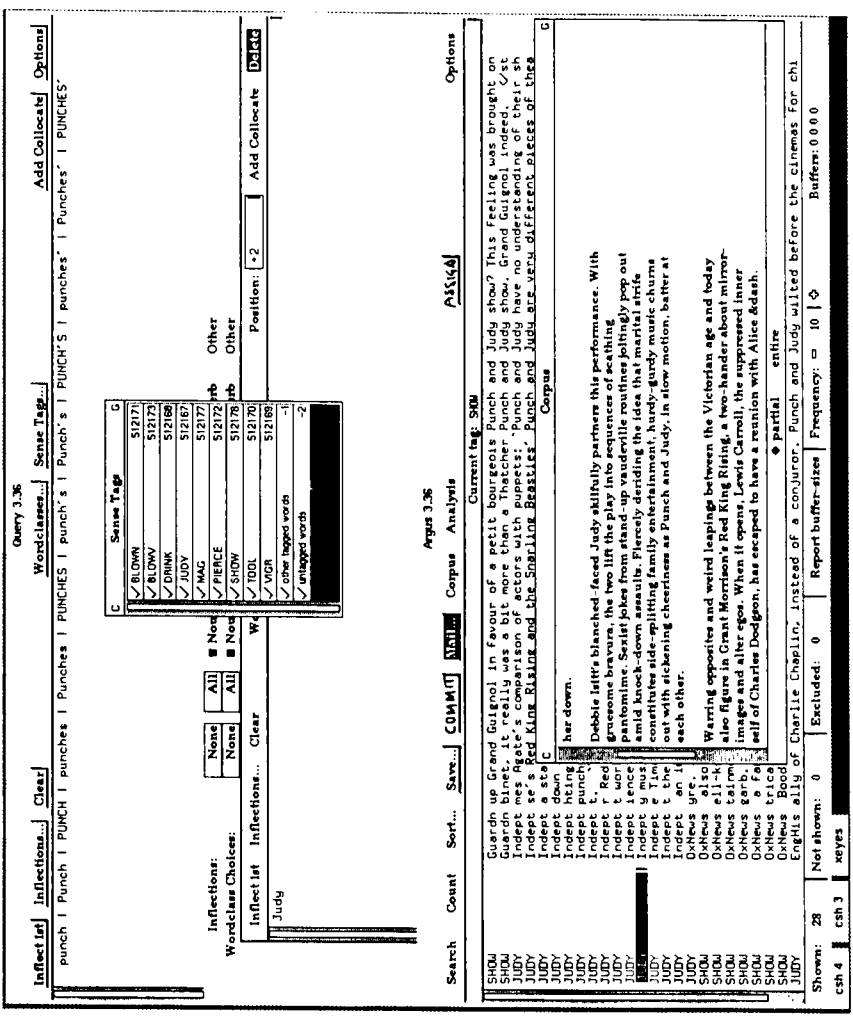


Fig. 14
(Argus) collocate *Judy*, and extended corpus context shown

In order to find a coherent citation I need to go back to the corpus for a fuller context. This is quickly done, by hitting the **Corpus** button in the Argus command line, and the corpus context is displayed (see Fig. 14). From the displayed corpus text, I block out the example I want, transfer the cursor to the Ajax screen and paste it into the ex field in the correct section. At this point, I number the senses in that section, which will be the third homograph headword, as its "Ord 3" numbering shows.

There will still be some corpus occurrences of 'Punch' which refer to the puppet, and others referring to the magazine. I decide to clear up that section of my draft dictionary entry, and return to the Argus screen. I reset the search conditions, this time looking for capitalized 'Punch' as a noun, with no specified context. (I kill the **Add Collocate** box containing 'judy', shown in Fig. 14 by hitting the **Delete** button.) I also change the "on/off" toggles on the Argus Sense Tags list (Fig. 15) so that Argus will display only occurrences that have not yet been sense-tagged. This facility to select according to the committed sense tags, as well as the other tagged features, greatly accelerates the lexicography. The output of this search is shown in Fig. 15.

The corpus reminds me of the existence of the phrase "as pleased as Punch". I add a sense frame to the Ajax entry, initialize the mnemonic 'pl', and tag the single relevant corpus line on display. However I suspect that in this phrase, 'Punch' may appear without its capital letter. I therefore set up an Argus search for the word 'pleased' in the left context of 'punch' (see Fig. 16, where the monitor colours identifying the "active" corpus line unfortunately show black in the screen dump) once again asking for untagged lines only, and finding only two, which I tag. I expand one to the corpus context and cut and paste an example for the draft entry.

With this little section of the entry nearing completion, I hit **Show** in the Ajax Commands menu, and bring it up in pseudo-dictionary format, moving it to the Atlas screen (see Fig. 17).

The bulk of the entry for *punch* (on the right of the screen) is as yet unformed, but a small part of it is gradually emerging. I leave the COD8 entry there, as the comparison is always useful.

Before moving on to compiling another part of the entry, I ask Argus to count the number of untagged noun-tagged occurrences remaining: 238. I sort these on right context and scroll through them, looking for inspiration for a new search, to no immediate avail; I re-sort them first on wordform and second on right context, and notice that there seem to be a lot of adjectives modifying *punch*. I sort again, this time on left context, and realize that the phrase "a punch" very often indicates the sense of "blow" (see Fig. 18). I set

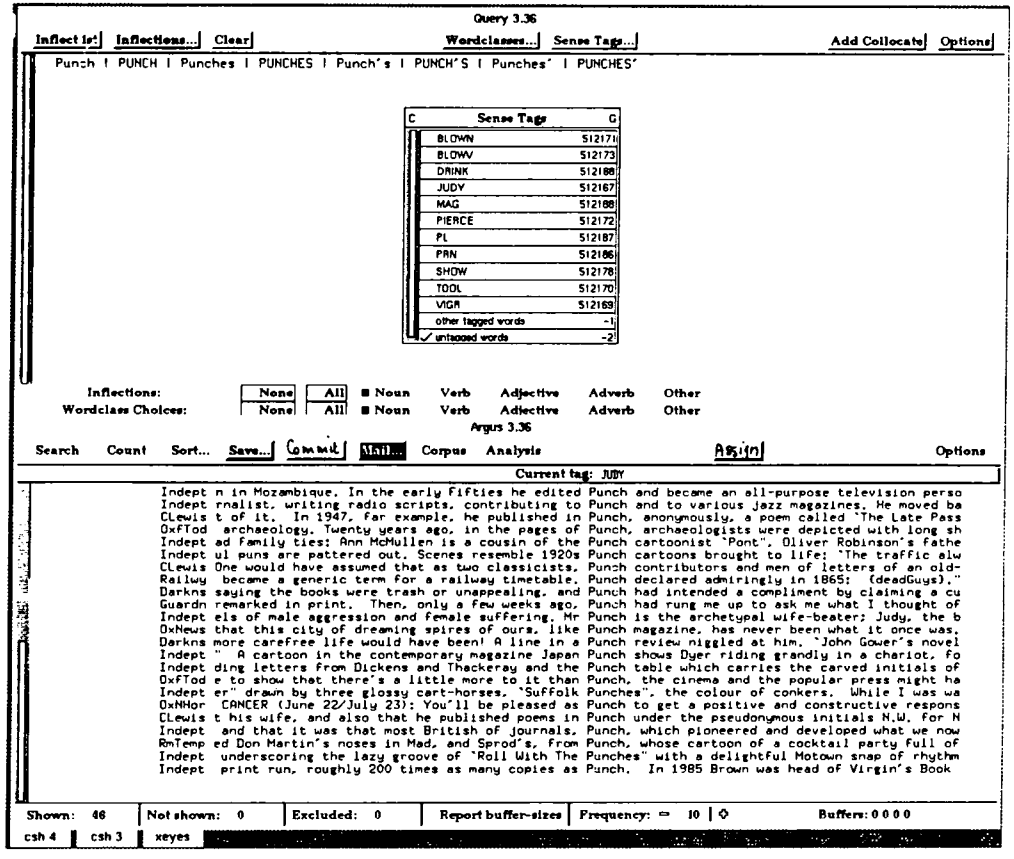


Fig. 15
(Argus) capitalized *Punch*, noun, non-sense-tagged lines only

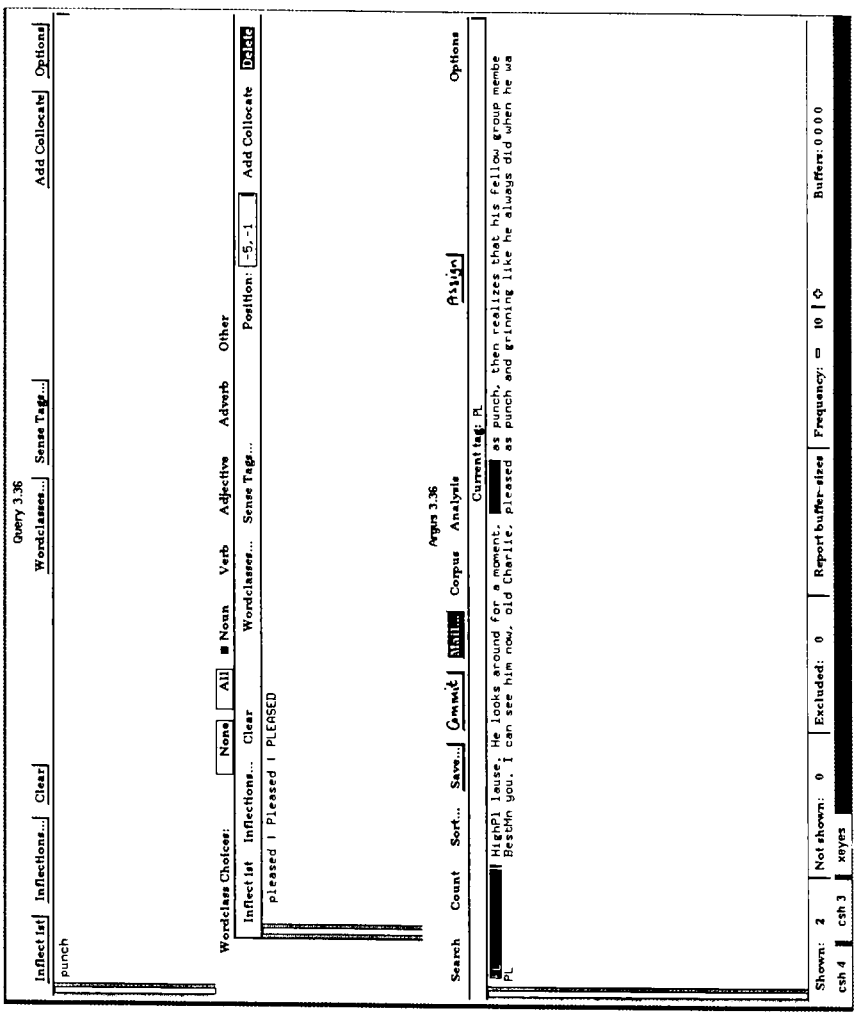


Fig. 16
(Argus) search for *pleased* in range –5,–1

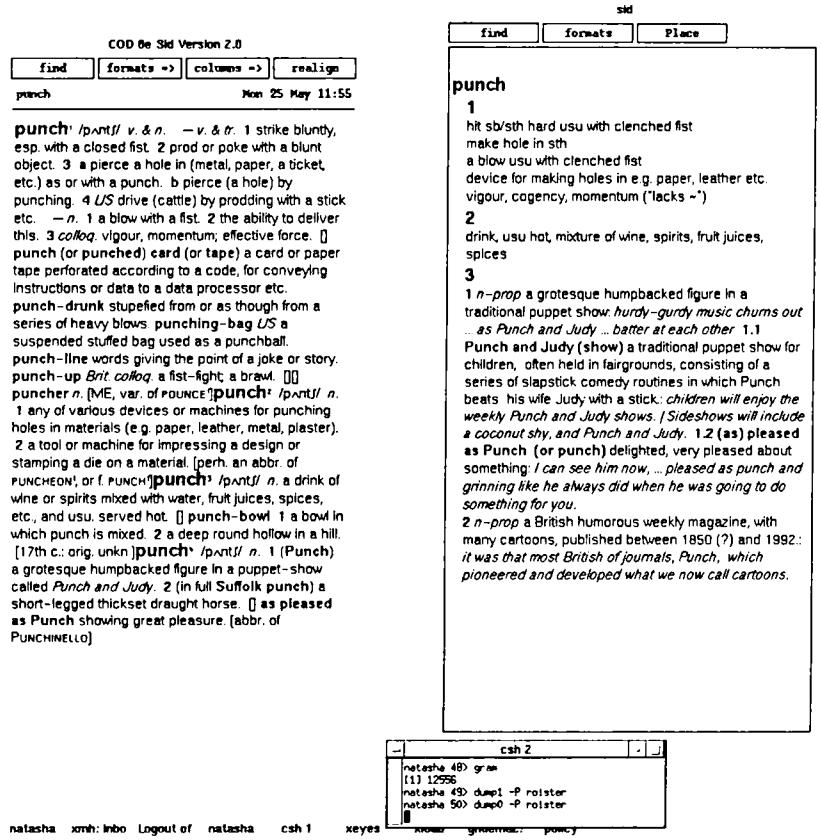


Fig. 17
(Atlas) COD8 entry (left) and embryonic new entry (right)

up the current tag as 'blown' and start assigning it to considerable blocks of concordances.

While doing this, I notice the phrases "to pack a punch" and "to throw a punch", and set these up as idioms in Ajax with the mnemonics 'pack' and 'throw'; the idiom "not to pull any (or one's) punches" is also recorded in Ajax with a mnemonic of 'pull'. Since 238 lines are too many to sense-tag individually, or even in blocks, I decide to set up more specific searches. In this way, I pull up 10 lines which I can block-assign as 'pull'. A similar technique accounts for 14 instances of "throwing punches" (all manifestly from commentaries on boxing matches, and I note 'Boxing' in the domain field of that section of the draft entry in Ajax). A third pass through the untagged corpus lines looking for wordforms of the verb *pack* in the immediate environment of *punch* produces 11 lines to tag with the 'pack' mnemonic (see Fig. 19, where the monitor's colour-highlighting of the specified collocate is blacking out the occurrences of *pack*).

Remembering a number of instances of *hot* collocating with the 'drink' sense of *punch*, I call them up, and tag six lines together. Checking back to the output of the **coll** command (see Fig. 5), I set up a search for collocates *pizza*, *bowl*, *party*, *rum* and *fruit*, and glean another score or so of tagged lines (see Fig. 20).

Beachcombing through the corpus offers at least the thrill of the chase. It occurs to me that one of the corpus works is all about someone trying to poison the punch that his wife is to drink; I resort the lines in corpus text order and manage to tag 24 occurrences in one block that way. Another of the texts in the corpus contains a discussion about the drink *punch*—another block to tag together.

As a bonus, sorting the corpus according to source text points up the fact that the four citations from *The Angler* magazine show the compound *bread punch*. I raise a new Ajax template by hitting **Another Ajax** in the Commands menu (see **3.3.1.1**), and keying a new (compound) headword 'bread punch' as a response to the "Which word?" query. A blank template appears. I insert a tag there (see Fig. 21), together with a brief definition and a note about the single source text where this usage appears, then hit the **Tags** toggle, transmitting it to Argus. Argus now has as live and valid sense tags all the *punch* mnemonics and the new *bread punch* mnemonic. I tag the four concordance lines in Argus with 'brp'.

I decide to look for occurrences of *punch* followed immediately by a preposition (in a hunt for "punch on the nose/jaw" etc.). The search conditions in Argus allow for search on wordclass tag alone, without any lexical item spec-

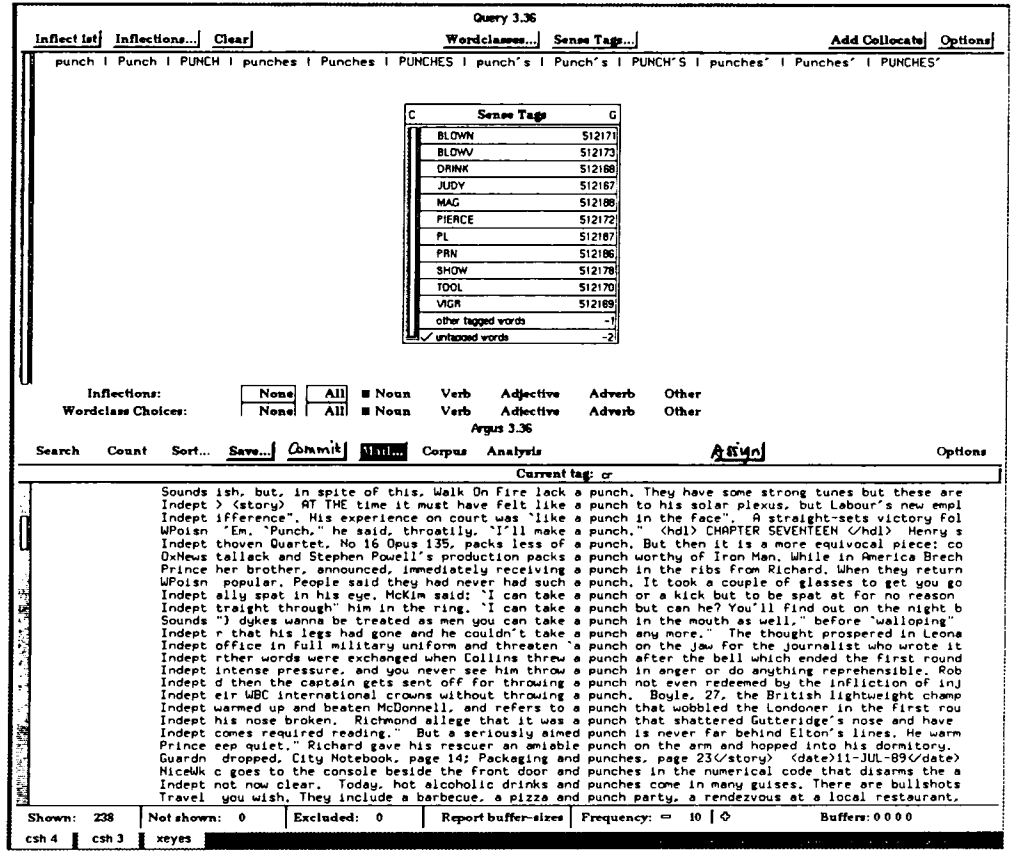


Fig. 18
(Argus) the pattern "a punch"; "Add Collocate" box is closed

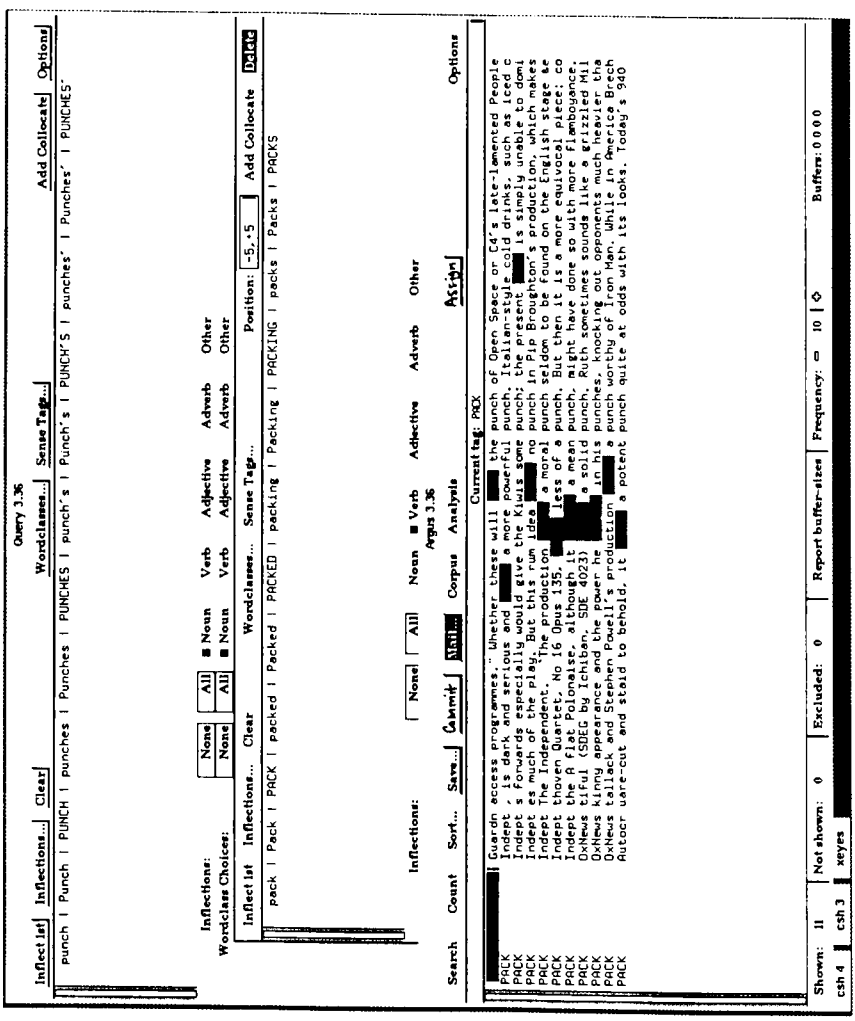


Fig. 19
(Argus) search for *pack* in range –5,+5

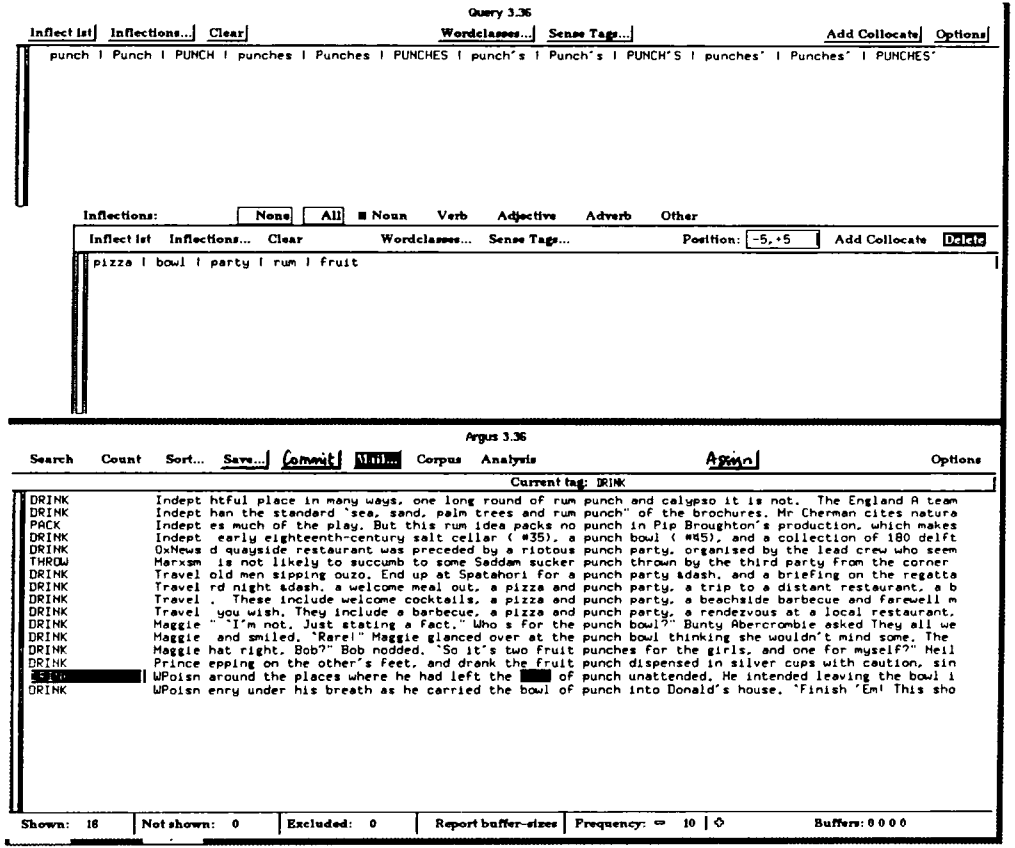


Fig. 20
(Argus) search for *pizza* or *bowl* (etc.) in range –5,+5

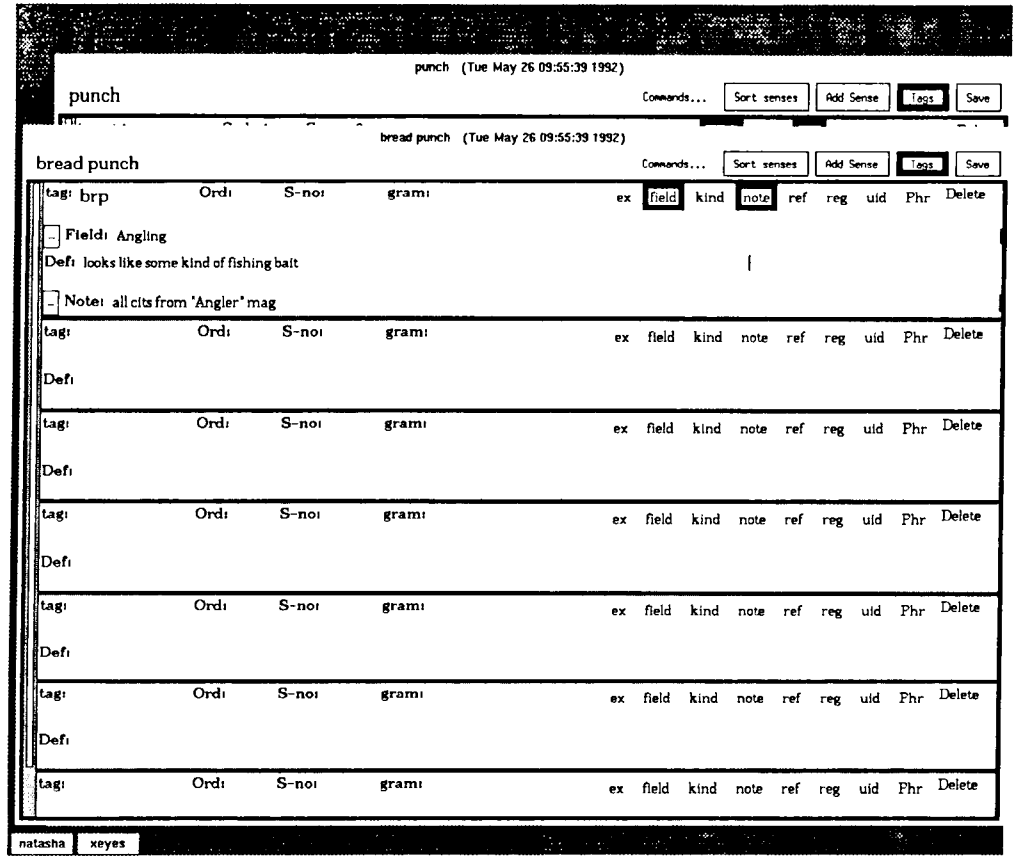


Fig. 21
(Ajax) new entry for compound headword *bread punch*

ified. This proves to be more fruitful, and 107 concordance lines (see Fig. 22) are found. Not all are correctly wordclass-tagged, but most are—the **Other** selection in the **Wordclass Choices** box pulls out everything that is not tagged Noun, Verb, Adjective or Adverb.

I realize that there are still many untagged noun lines like "a punch in the face" or "a punch on the jaw". I therefore set up a search condition in Argus to pull out untagged concordances where *in* or *on* occurs within four words after *punch*, and Argus finds 30 matching lines, most of which I can sense-tag 'blown' (= "blow: noun") quickly, in blocks.

Wishing to tag the corresponding verb lines, I set up a search condition in Argus to pull out untagged concordances where *in* or *on* occurs within four words after *punch* verb occurrences, and Argus finds 55 matching lines (see Fig. 23). It is clear that some semantic feature tagging on corpus nouns (e.g. BODYPART) would allow me to pull out large numbers of "assault" and "boxing" occurrences of *punch*, both noun and verb.

Meanwhile, I notice other senses of *punch* for which I have not yet got tags established, e.g. "can be punched into the numeric keypad", for which I set up a new mnemonic. The compound *punch(ed) card* requires a separate headword entry in the database; this is made, and this compound is given a mnemonic.

At this point Argus is working on 26 active sense-tags, belonging to three headwords all concurrently active in Ajax (*punch*, *punch(ed) card* and *bread punch*). The tags are as follows shown in Fig. 24.

I set up the condition "Untagged lines" only, and hit **Count** in Argus, to learn that 223 lines remain to be tagged (see Fig. 25).

Halfway through the tagging process the lexicographer's job satisfaction is at its nadir. (And *punch* is not a highly frequent word.) I scroll through these lines, looking for inspiration. How can I pull up large blocks of the same sense? I wish Argus could already find possessives and pronouns, since this would bring up all the lines like "punched her on the nose" and "punched him". I note with vague interest how often someone's name occurs as the object of the "assault" sense (mainly from newspaper texts, of course), and wish Argus could search on proper nouns, which, although tagged as a class, are at this moment still subsumed under Other wordclasses. I think that if we already had the semantic features tagged on nouns I could pull up all the +HUMAN nouns after *punch* and collect a whole senseful of citations that way.

I call up all the lines with hole as a collocate, and tag these, and similarly all the *punch(ed) card* concordances. Moving back and forth between Argus and Ajax, I finally complete the tagging procedure, adding en route headword

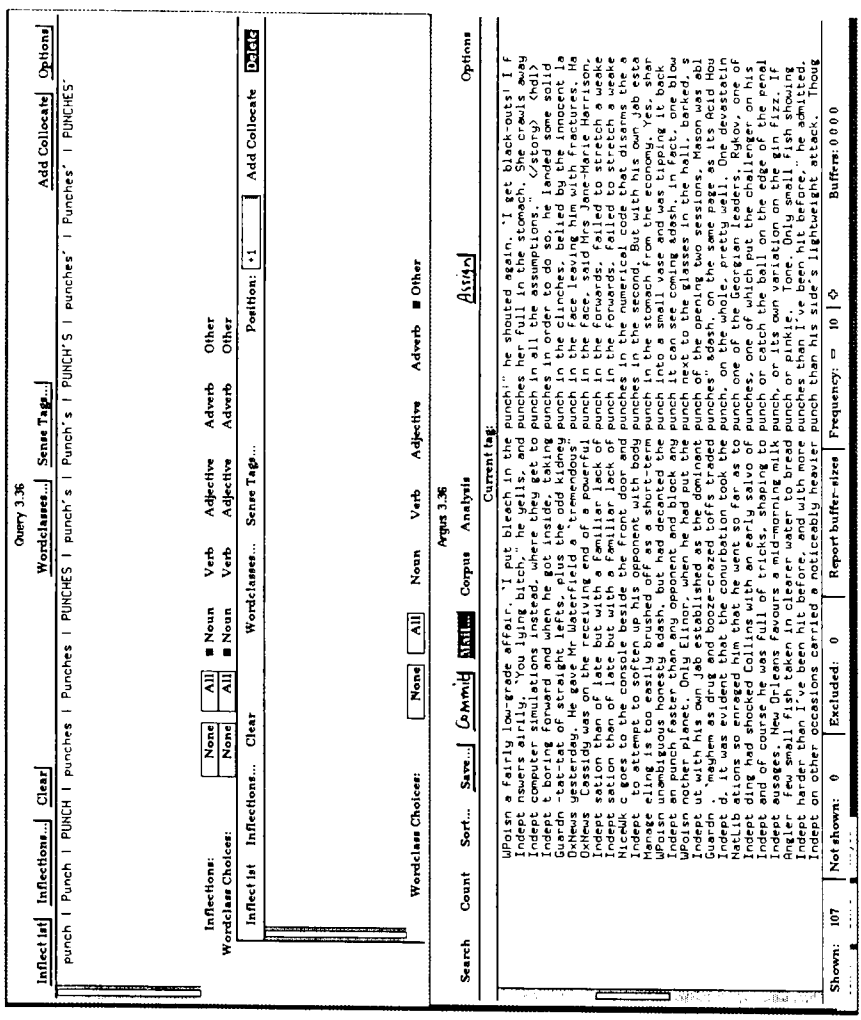


Fig. 22
(Argus) search for "Other" wordclasses at position +1

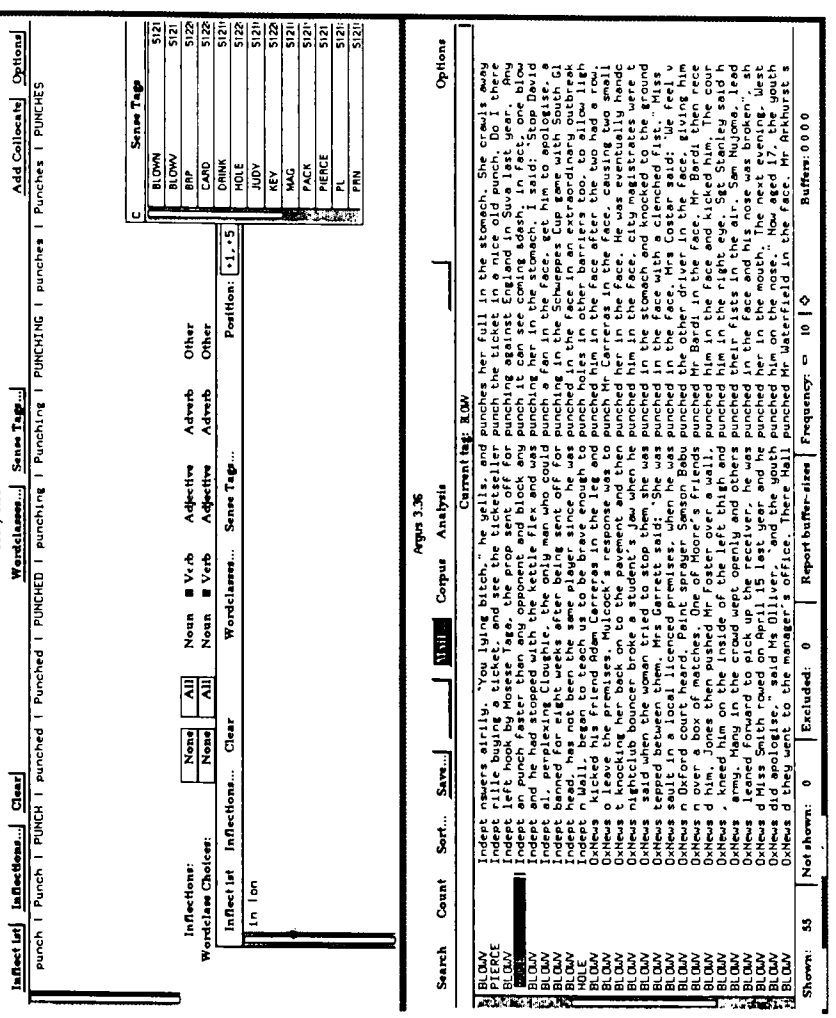


Fig. 23
(Argus) *punch* verb with *in* or *on* in Range +1,+5
showing assigned sense tag "blowv"

MNEMONIC SENSE TAGS

for senses in entry shown in Fig. 35

tag	sense	no.
AIR	*punch the air*	1.2
BLOWN	*blow - noun*	4
BLOWV	*blow - verb*	1
BRP	*bread punch*	headword
DATA	*punch (in) data*	2
DRINK	*fruit punch*	punch[2]
FIST	*punch one's fist through sth*	1.1
HOLE	*punch a hole in sth*	3
JUDY	*Punch the puppet*	punch[3] 1
KEY	*punch a key on keyboard*	2.1
LINE	*punch line*	headword
MAG	*Punch the magazine*	punch[3] 2
PACK	*pack a (powerful) punch*	4.4
PAPER	*punch ... a paper bag*	1.3
PIERCE	*punch a ticket*	3.1
PL	*pleased as Punch*	punch[3] 1.2
PULL	*doesn't pull any punches*	4.5
SHAPE	*punch out a hexagon*	3.2
SHOW	*Punch & Judy show*	punch[3] 1.1
SUFF	*Suffolk punch (horse)*	headword
SWAP	*swap punches*	4.2
TAPE	*punch(ed) tape*	headword
THROW	*throw a punch*	4.1
TOOL	*tool for making holes in sth*	6
TRADE	*trade punches*	4.3
VIGR	*lacks punch*	5

Fig. 24
List of "active" mnemonic sense tags for *punch*, *punch(ed) card* and *bread punch*

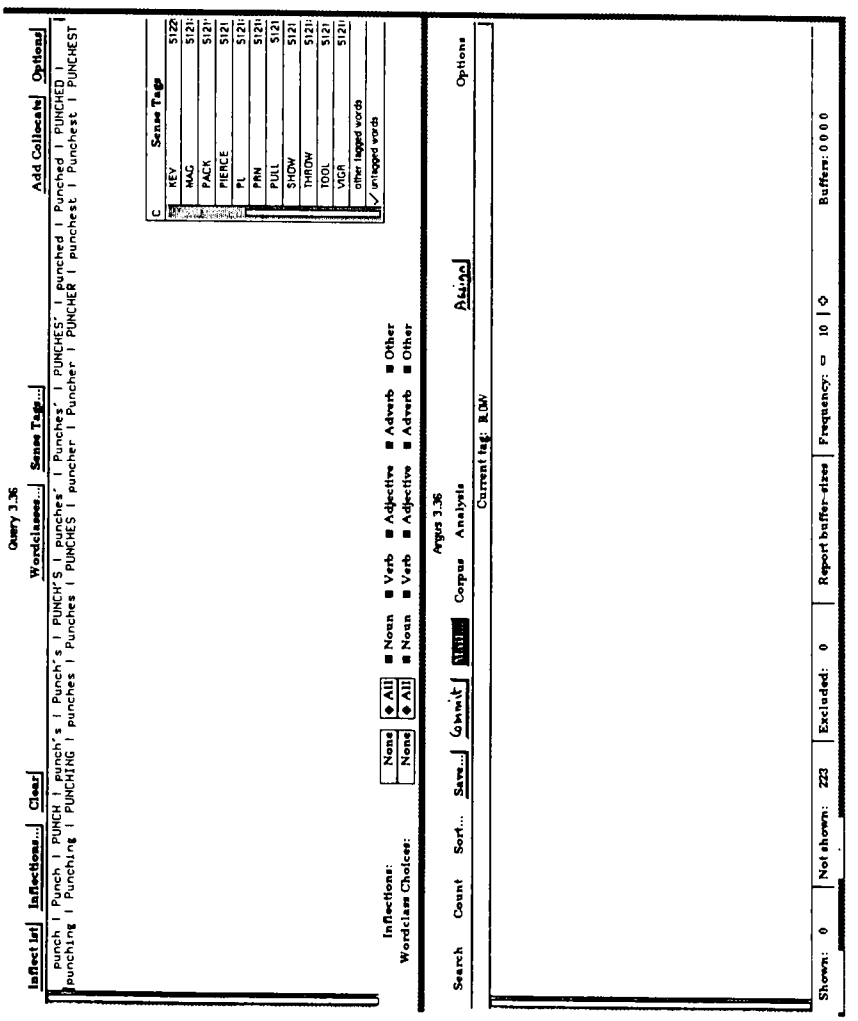


Fig. 25
(Argus) response to search for all lines still without sense tags

entries for *punched tape*, *punch-up* and *punch(-)drunk*, bringing the sense-tag count up to 29.

4.3. From draft entry to final version

Once all the relevant corpus lines are tagged in Argus, the dictionary entry in Ajax must be completed. At present the draft entry is extremely rough and ready (see the extract in Fig. 26).

All the senses are there, with their mnemonic tags, but there is no coherent ordering, and few if any completed definitions. Almost all the dictionary senses require exemplification, and most of the other information (register and domain labels, cross-references etc.) is missing. Entries for the currently active related compounds with headword status have been iconified and lie along the base of the screen.

The first task is to order the senses in Ajax, to get them as near as possible to what will be the final dictionary-database order. This is where the **Show** function in the **Commands** menu comes into its own: an entry like *punch*, which (with related compound headword entries) has 29 active tags, will overflow the Ajax screen many times over. It is vital to be able to see the whole entry during the next part of the process. This is achieved by means of the **Show** command.

First, however, I make an initial attempt at numbering the senses and ordering them within the entry—or, rather, entries, for the "Ord" numbers in Ajax are used to distinguish homograph headwords. The provisional policy for this database is to treat as separate headwords homographs which are cognitively discrete, those for which an act of scholarhip is needed to link them etymologically or in any other way. Thus, the *punch* group will have three homographs (the main noun/verb, the drink and the Punch and Judy character).

Before I start on systematic sense numbering, I have to have some idea of what the contents of the entry are. I hit **Show** in the **Commands** menu of Ajax, and a pseudo-dictionary entry appears (see Fig. 27).

The three main homographs stand out clearly: the third is beginning to look reasonably solid, but the second lacks everything but a definition, and the first is a jumble of odd facts. Since the sense-numbering in Ajax is not complete (as is apparent in Fig. 27), there are many sense numbers missing from the first major section (number 1) in the pseudo-print format; however, the senses themselves are there, as may be gleaned from a careful study of the content of the draft entry in Fig. 27.

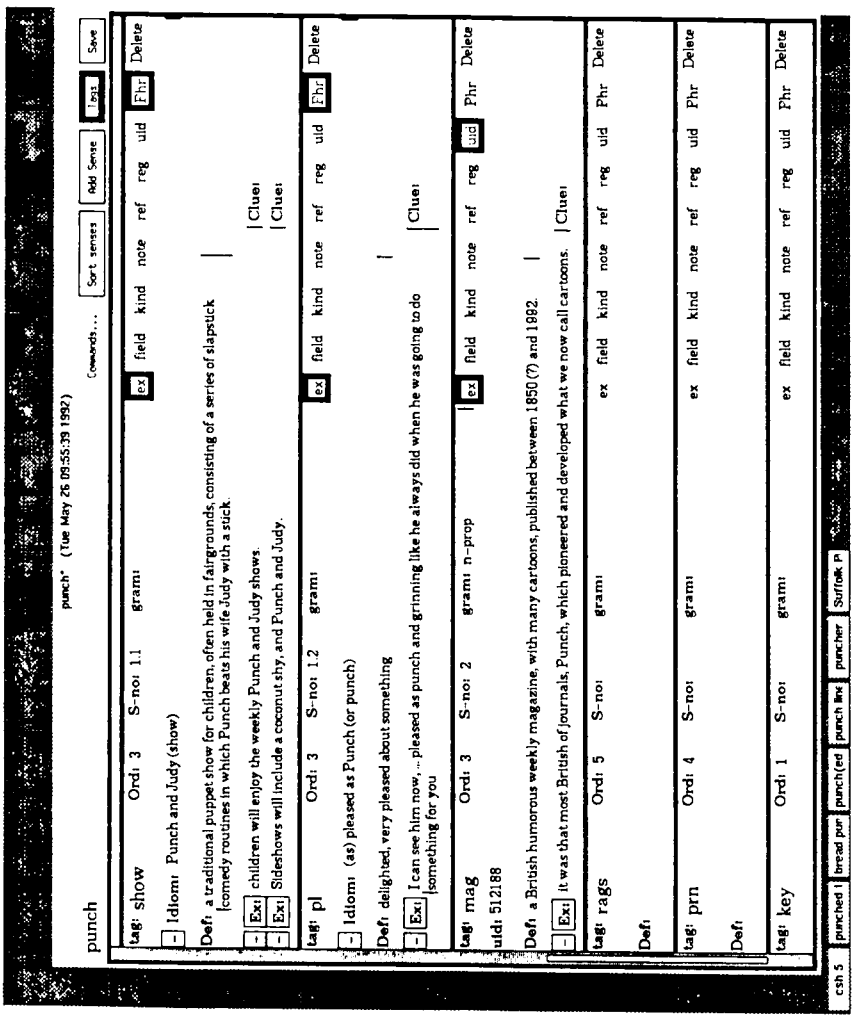


Fig. 26
(Ajax) extract from *punch* early draft entry

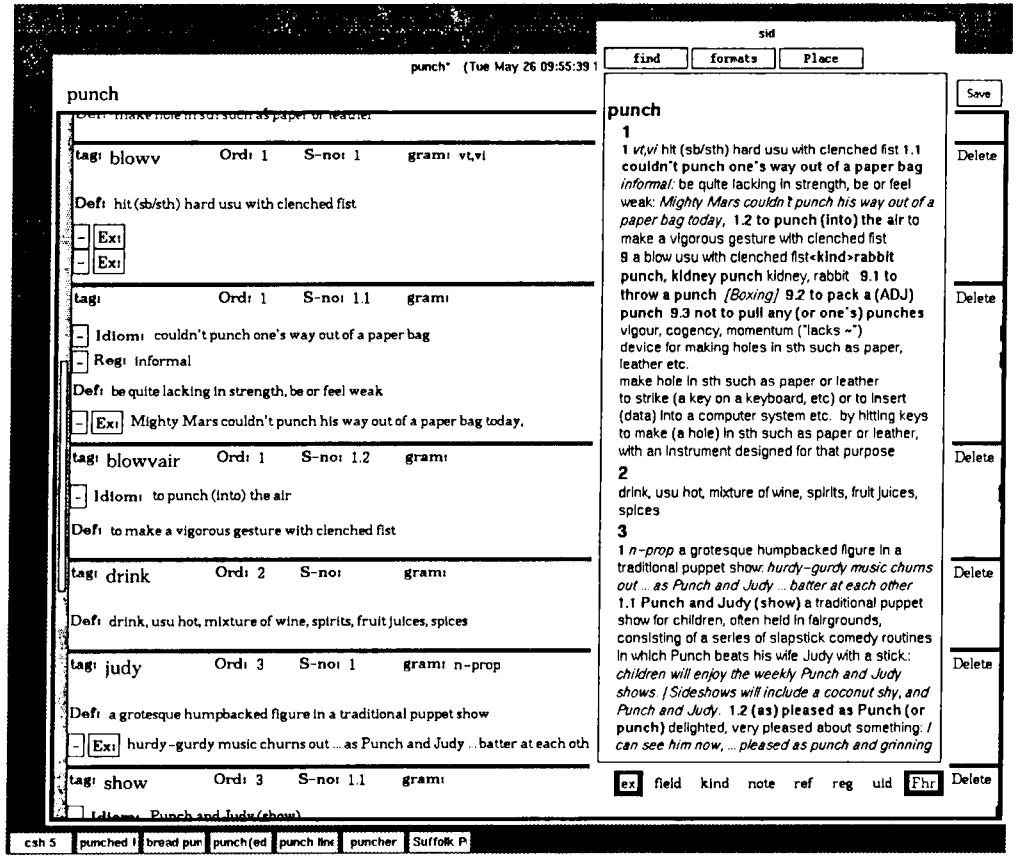


Fig. 27
(Ajax) early draft entry in Ajax and print format

I use this overview to help me order the senses, and I also key in "cod8 punch" to bring the COD8 entry up in the Atlas screen, for reference. (I note in passing that COD8 marks *punch-up* as British English, and I add that to my draft entry for this word.)

I number the verb senses and call up another **Show** version (see Fig. 28).

The verbs outline seem acceptable, so I go back into the Ajax entry and number the noun senses, calling it up once again with Show to check that it is complete.

Now that I am satisfied (for the moment) with the numbering system, I reorder the senses in Ajax by hitting the **Sort Senses** button in the top command line, to make the Ajax template sense order correspond to the actual numbering in the entry. This is so that I can work down through the entry in Ajax, going back and forth into Argus to collect appropriate examples, and make sure that the facts in the Ajax entry correspond to the corpus data. Figure 29 shows part of the draft entry in Ajax format, and also a larger section from it in the print format.

The work remaining to be done on fleshing out the senses in the draft entry is essentially repetitive. You work through screen after screen of material in Ajax: scan the draft definition; call up the corpus lines tagged for that sense; select typical and informative examples (for this dictionary-database, provisionally at least, length restrictions are relaxed); check for lurking idioms and collocations which have eluded you so far; polish the definition; insert the grammatical notation; insert labels relating to subject fields, register, style etc; and add any cross-references that need to be recorded. This is a provisional and very basic dictionary entry—other material, such as pronunciations, etymologies and so on, will be added later, when the dictionary style and content are defined in more detail, omissions are repaired, and the entry is refined and polished to bring it up to standard.

I therefore start on Sense 1 ("hit sb/sth hard, usu with clenched fist"), setting up a search in Argus for all the lines tagged with the mnemonic 'blowv' (= "blow: verb") and being offered 176 concordances (see Fig. 30) to scan in my search for the ideal example—one that shows the headword in a very typical context, that exemplifies constructions commonly associated with the word, and yet is informative enough in content to flesh out the definition and make the meaning clearer to the dictionary user.

Since *face* is one of the statistically significant collocates of *punch* (as was shown by the **coll** command, see Fig. 5), and since I have noted so many lines displaying the frame "X punched Y in the BODYPART", I choose a line that exemplifies both of these features. The displayed KWIC concordances

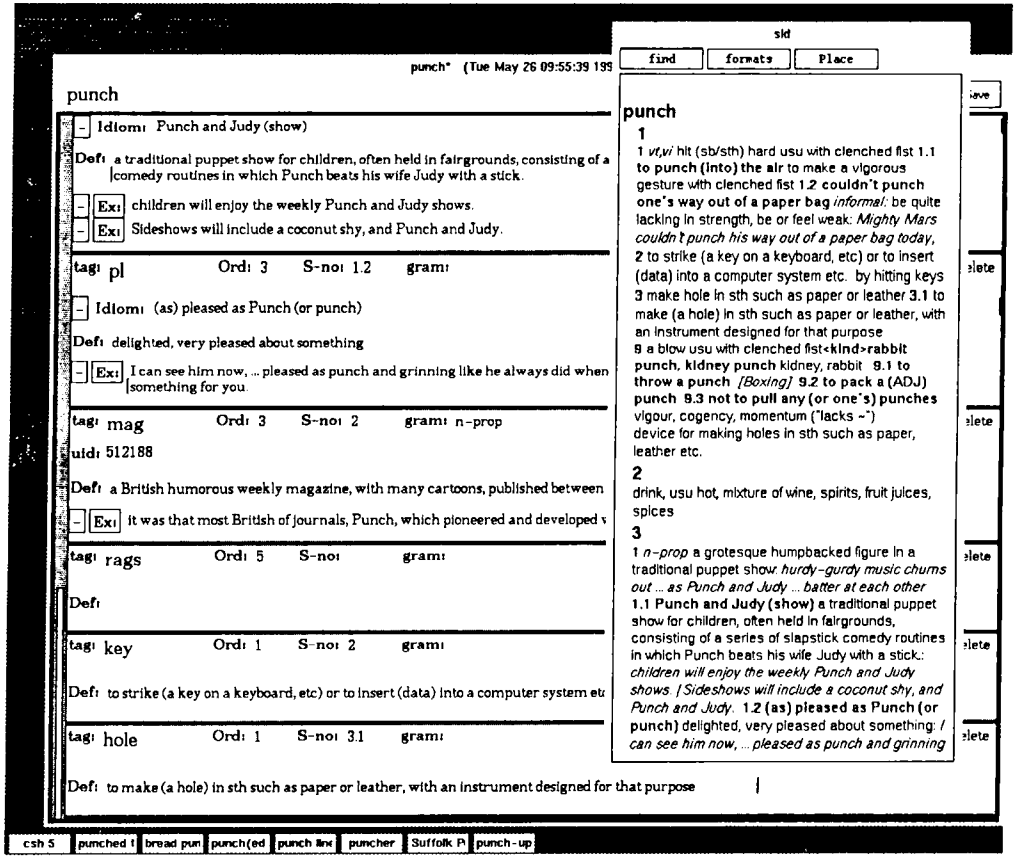


Fig. 28
(Ajax) template and print versions of draft entry, verb senses numbered

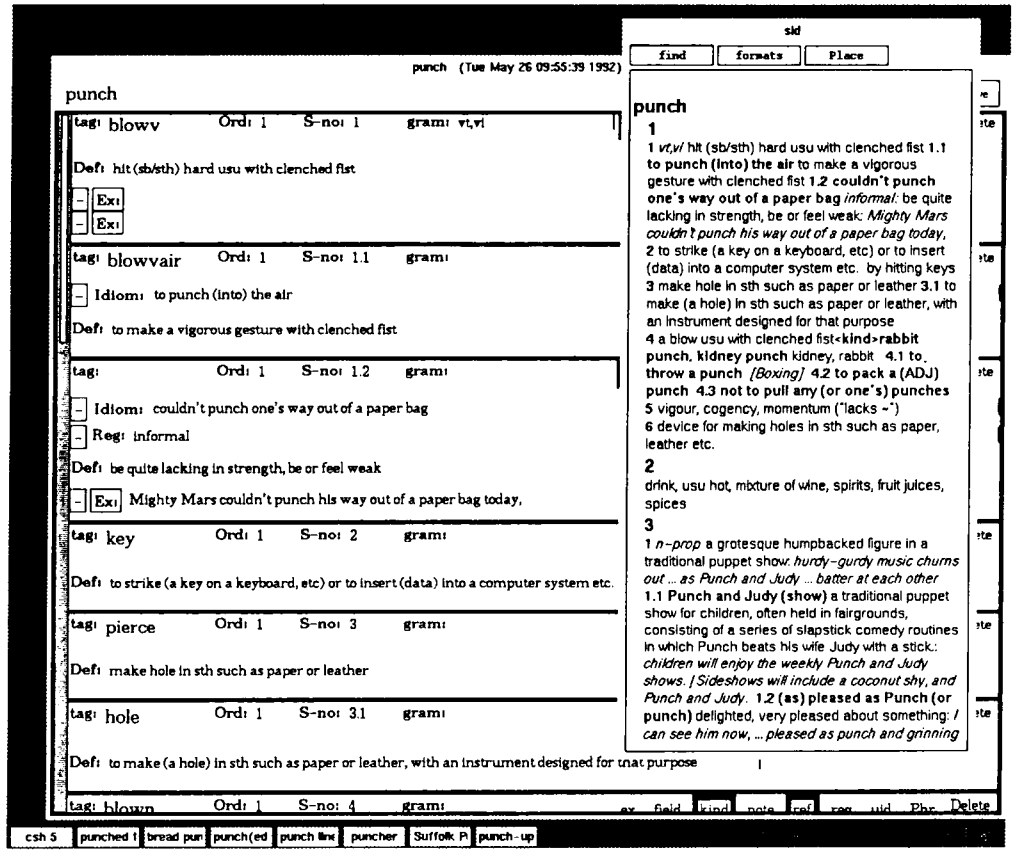


Fig. 29
(Ajax) template and print versions of draft entry, verb and noun senses numbered

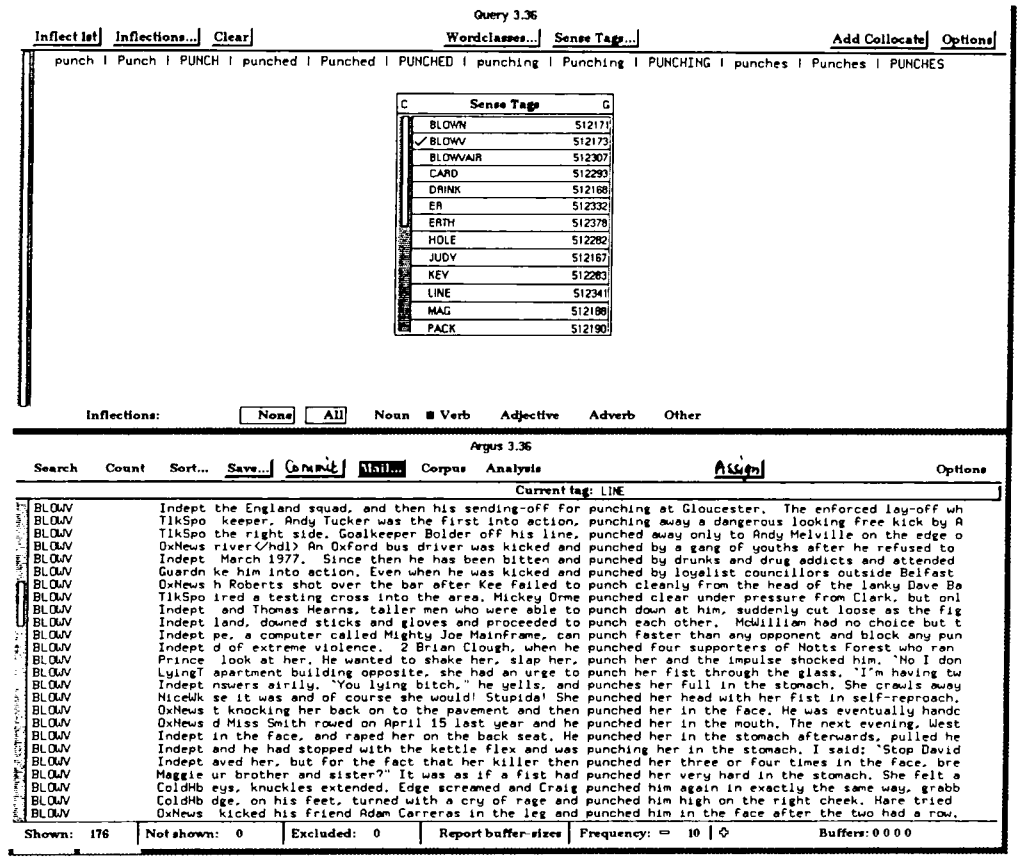


Fig. 30
(Argus) citations for verb sense 1 selected on tag 'blowv'

are inadequate, and I have to look at the larger context within Argus (see Fig. 31), by hitting the **Corpus** button, and to cut and paste the citation from the corpus into the Ajax entry, Sense 1.

In passing I note a couple of corpus lines with the collocation "punched his fist through . . . " and decide to add that as a distinct collocation in the entry. This means adding a sense frame in Ajax and extracting an example from Argus. I decide that it does not require a definition, its context in the entry making it quite transparent. However, it seems sensible to reorder the idioms within Sense 1. I change the numbers in the Ajax draft, which immediately changes the layout in the output from the **Show** command (see Fig. 32), but it does not seem worthwhile to reorder the Ajax entry on the screen.

Looking for examples of "to punch (into) the air" I change the sense-tag mnemonic in the Argus sense conditions, find one, and cut and paste it into the Ajax entry. This process continues until the verb section of the entry is complete. I pull it up with the **Show** command (see Fig. 33) just to check it, before moving on to the noun.

I study the draft noun section in the **Show** print format (see Fig. 34).

This time, when I go back to the corpus for examples, I sort them on context to the left of the target word. This helps me find an example showing another significant collocate *powerful*, and also the plural use of the noun.

Finally, the entry is as complete as I wish to make it for the purpose of this demonstration, although the definitions clearly need refining, and some senses, and examples, are missing from a comprehensive description of the word. A command prints it off in some semblance of a dictionary entry (see Fig. 35).

5. Conclusion

I believe we have all learnt a great deal from this collaboration. For me, it was particularly interesting (and often chastening) to see how a new routine that seemed a stroke of genius at the drawing-board stage simply complicates the lexicographical process too much.

The task of sense-tagging all the occurrences of the headword in the corpus proved very labour-intensive, despite the powerful tools. Compiling a database entry for a word with six or seven hundred occurrences, and sense-tagging these, took a couple of days at least.

However, much effort was put into making the programs run faster, and the interface more lexicographer-friendly. The "task list"—a wish list which the lexicographers of the team drew up, and which was systematically im-

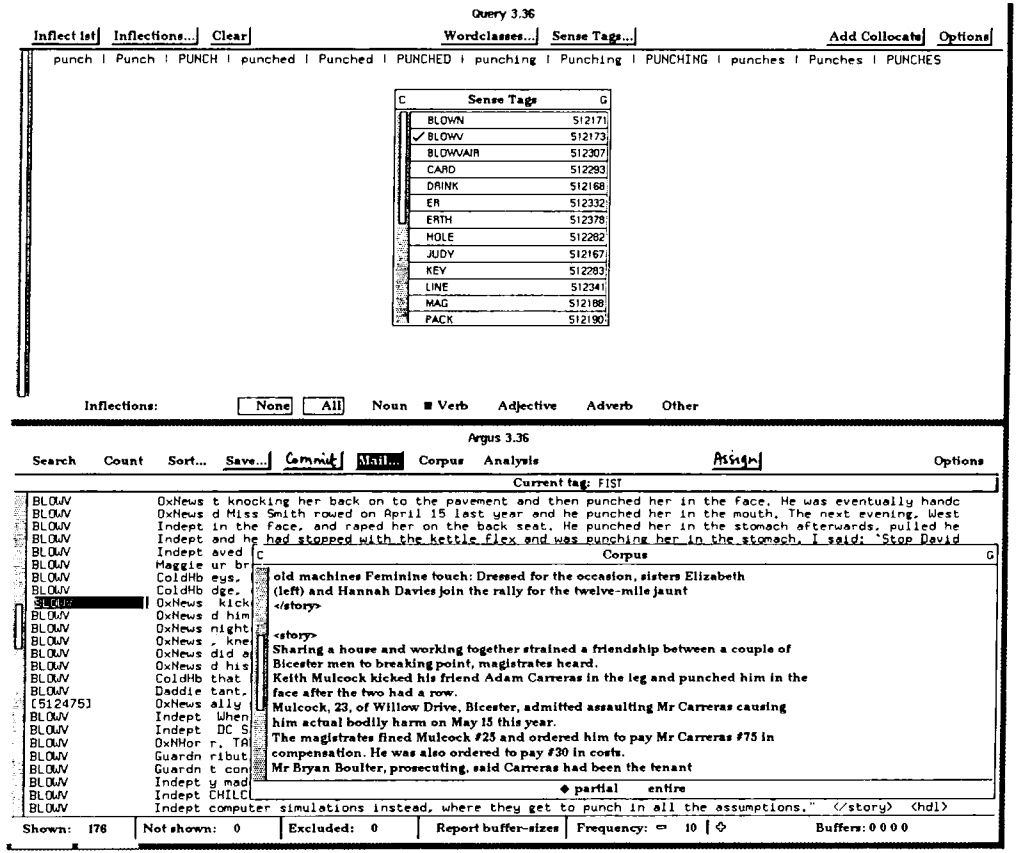


Fig. 31
(Argus) expanded context for "punched him in the face"

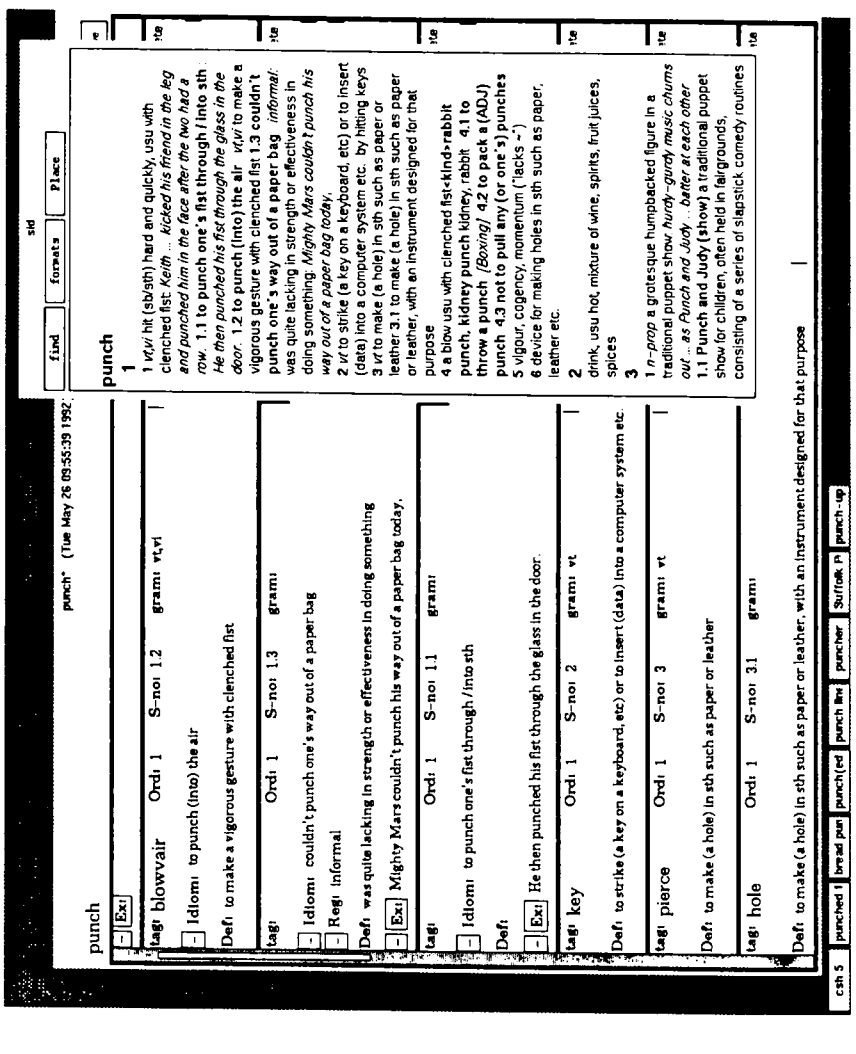


Fig. 32
(Ajax) draft: template and print version from the "Show" command which automatically reorders the senses

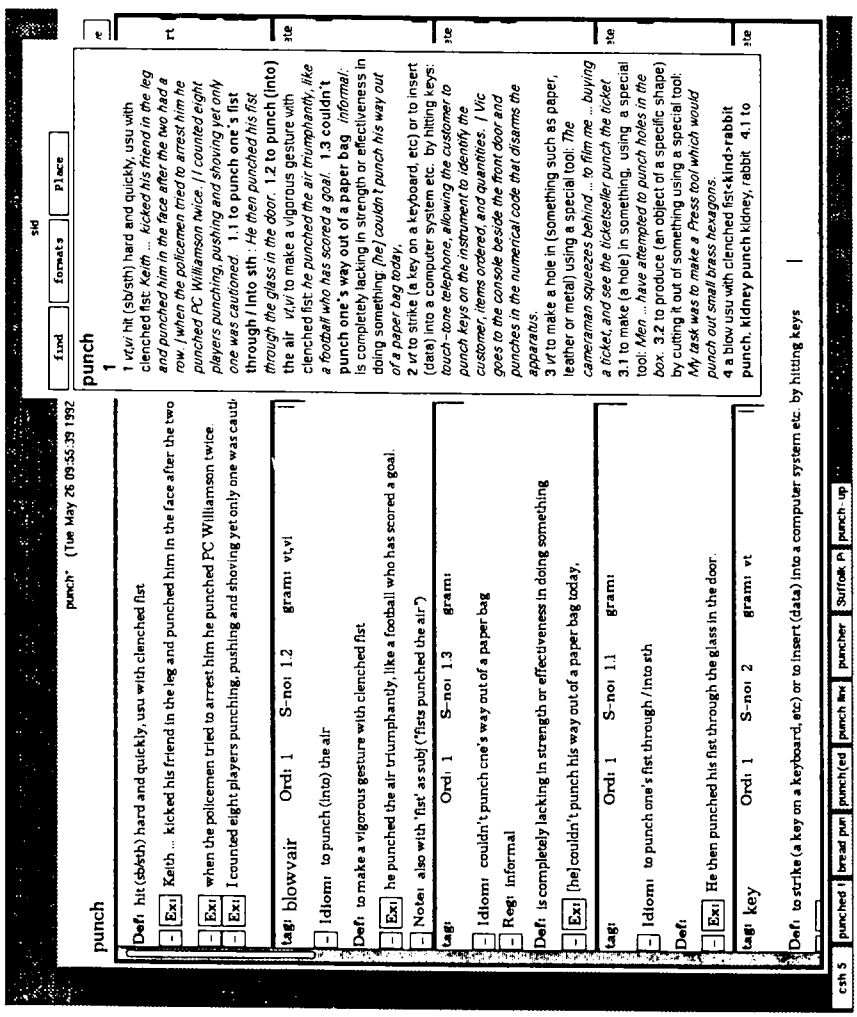


Fig. 33
(Ajax) draft: template and print version with section complete

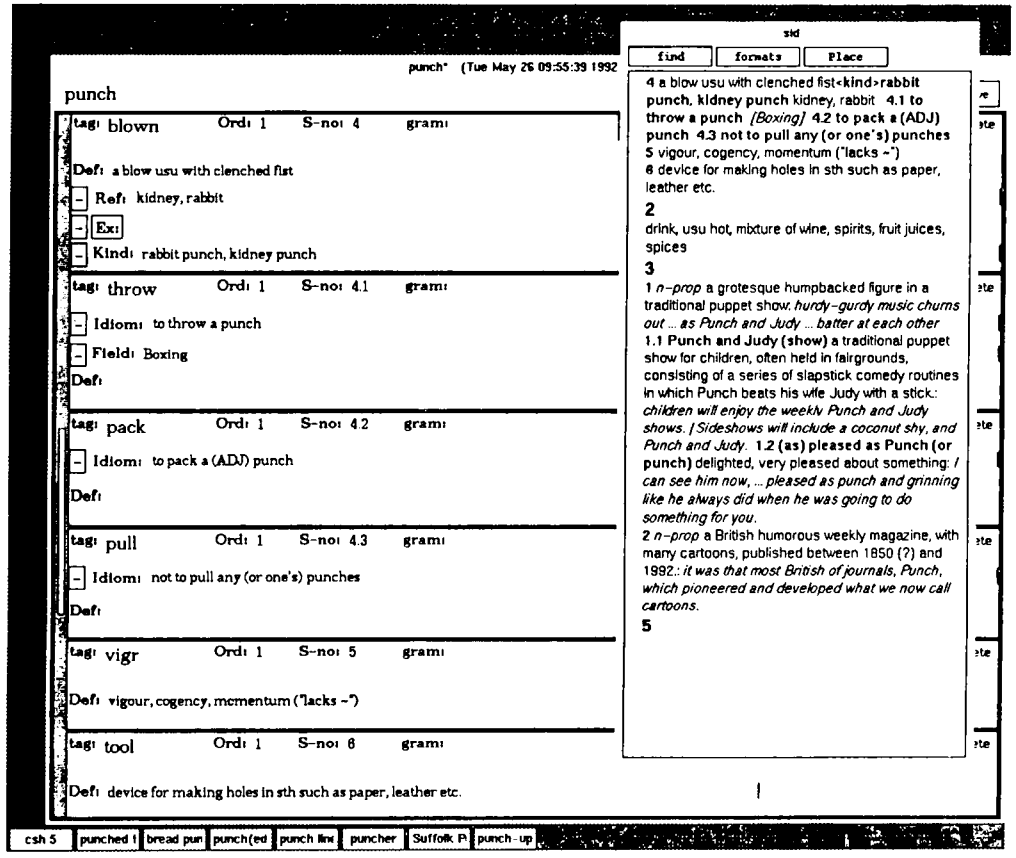


Fig. 34
(Ajax) print version shows sparse noun section

punch

punch [1]

1 *vt,vi* to hit (someone or something) hard and quickly with a clenched fist:
Keith ... kicked his friend in the leg and punched him in the face after the two had a row. || *when the policemen tried to arrest him he punched PC Williamson twice.* || *I counted eight players punching, pushing and shoving yet only one was cautioned.*

1.1 to punch one's fist through / into something
He then punched his fist through the glass in the door.

1.2 to punch (into) the air*vt,vi* to make a vigorous gesture with a clenched fist:
he punched the air triumphantly, like a football who has scored a goal.

1.3 couldn't punch his / her way out of a paper bag*informal.*is completely lacking in strength or effectiveness :
[he] couldn't punch his way out of a paper bag today.

2 *vt* to enter (numbers or other data) into a computer or other machine by striking keys on a keyboard:
Vic goes to the console beside the front door and punches in the numerical code that disarms the apparatus.

2.1 *vt* to strike (a key on a keyboard) in order to insert data into a computer or other machine:
touch-tone telephone, allowing the customer to punch keys on the instrument to identify the customer, items ordered, and quantities.

3 *vt* to make (a hole) in something, especially by making use of a special tool for the purpose:
Men ... have attempted to punch holes in the box.

3.1 *vt* to make a hole in (something such as a ticket) using a special tool:
see the ticketseller punch the ticket.

3.2 *vt* to produce (an object of a specific shape) by cutting it out of something using a special tool:
My task was to make a Press tool which would punch out small brass hexagons.

4 *nc* [Boxing, Fighting]a blow with a clenched fist:
Stanley came at me and knocked me down with a powerful punch. || *the prosecution had not proved that it was a deliberate punch and not just an accidental blow.* || *As a welterweight he proved to be less destructive; the result ... of larger men being better able to withstand the impact of his punches.*

4.1 to throw a punch[Boxing]to aim a punch at one's opponent:
further words were exchanged when Collins threw a punch after the bell which ended the first round and Laing retaliated by appearing to aim a kick at the challenger.

4.2 to swap punches*British.*to become involved in a bout of fisticuffs:
opposition MPs swapped punches and shoved over elderly members of the ruling Nationalist Party... after a disputed committee decision.

4.3 to trade punches*American.*= to swap punches

4.4 to pack a (powerful (etc.)) punchto be very effective, successful, and impressive (used of a performance, work of art, artifact etc.):
The ... Beethoven Quartet, No 16 Opus 135, packs less of a punch. || *The espresso, in small cups, is dark and serious and packs a more powerful punch.* || *Volvo's turbocharged big saloon ... packs a potent punch quite at odds with its looks.*

4.5 not pull any (or one's) punchessay something critical in a direct way, without trying to soften the impact of the words:
The young reporter pulled no punches when it came to direct questions. || *Voinovich's satire pulls no punches and hits where it hurts most.*

5 *nu* brevity and effectiveness (used of a person, performance, or the way something is told or carried out):
... communications must have clarity and cohesion. Above all they must have punch - beware of flogging an issue too hard. || *Much British cinema does lack emotional punch.*

6 *nc* a tool or machine for making holes in something such as paper, leather, metal, etc.:
showing me how the punch fitted into the guide plate

punch [2]

nu,nc a mixture of fruit juices and spices, often with wine and spirits, made usu for a party, and sometimes drunk hot:
Fruit punch was served with the meal . || *As winter drew on, many a pleasurable hour was spent in front of a blazing coal fire drinking mulled concoctions and hot punches.*

punch [3]

1 Punch*n-prop* the principal male character in a traditional puppet show, a grotesque hump-backed figure:
hurdy-gurdy music churns out ... as Punch and Judy ... batter at each other

1.1 Punch and Judy (show)a traditional puppet show for children, often held in fairgrounds, consisting of a series of slapstick comedy routines in which Punch beats his wife Judy with a stick.:
children will enjoy the weekly Punch and Judy shows. || *Sideshows will include a coconut shy, and Punch and Judy.*

1.2 (as) pleased as Punch (or punch)delighted, very pleased about something, (sometimes used to imply smugness):
I can see him now, ... pleased as punch and grinning like he always did when he was going to do something for you.

2 *n-prop* a British humorous weekly magazine, with many cartoons, published between 1850 (?) and 1992:
it was that most British of journals, Punch, which pioneered and developed what we now call cartoons.

Fig. 35.

Draft entry output from "printentry punch" command in Atlas

plemented by the computer scientists—grew longer and longer.The software improved and developed from week to week. This is no more than a status report, as at May 1992. The Hector project ended in March 1993.

References

Atkins, B.T.S. 1987. Semantic ID tags: Corpus Evidence for Dictionary Senses. In: The Uses of Large Text Databases, Proceedings of the Third Annual Conference of the UW Centre for the New OED, Waterloo, Canada.

Black, E. 1988. An Experiment in Computational Discrimination of English Word Senses. IBM Journal of Research and Development, Vol. 32, No. 2, IBM, Yorktown Heights, NY.

Byrd, R.J-Calzolari, N.-Chodorow, M.-Klavans J.-Neff M.-Rizk, O. 1987. Tools and methods for Computational Lexicology. In: Computational Linguistics 13: 219–40.

Calzolari, N.-Picchi, E. 1988. Acquisition of Semantic Information from an On-Line Dictionary. In: Vargha, D. (ed.): Proceedings of COLING 88 Budapest. J. von Neumann Society for Computing Sciences, Budapest, Hungary.

Chodorow, M.S.-Byrd, R.J.-Heidorn, G.E. 1985. Extracting semantic hierarchies from a large on-line dictionary. In: Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, 299–304.

Church, K.-Hanks, P. 1990a. Word association norms, mutual information, and lexicography. In: Computational Linguistics 16:1.

Church, K.-Gale, W.-Hanks, P.-Hindle, D. 1990b. Using statistics in lexical analysis. In: Zernik, U. (ed.): Lexical Acquisition: Using On-Line Resources to Build a Lexicon. Lawrence Erlbaum Associates, NJ.

Church, K.-Gale, W.-Hanks, P.-Hindle, D.-Moon, R. 1994. Lexical Substitutability. In: Atkins, B.T.S.-Zampolli, A. (eds): Computational Approaches to the Lexicon. Oxford University Press, Oxford.

Clear, J. H. (1993). Corpus sampling. In: Leitner, G. (ed.): New Dimensions in Corpus Linguistics: Proceedings of the 11th ICAME Conference, Berlin, June 1990. de Gruyter, Berlin.

Glassman, L.-Grinberg, D.-Hibbard, C.-Meehan, J.R.-Guarino Reid, L.-van Leunen, M-C. 1992. Hector: connecting words with definitions. In: Screening Words: User Interfaces for Text. Proceedings of the Eighth Annual Conference of the UW Centre for the New OED and Text Research, 37–74. University of Waterloo, Waterloo, Canada.

Johansson, S.-Hofland, K. 1989. Frequency Analysis of English Vocabulary and Grammar. Oxford University Press, Oxford.

Lesk, M. 1986. Automatic sense disambiguation using machine-readable dictionaries: How to tell a pine cone from an ice cream cone. In: Proceedings of SIGDOC1, 24–6.

Levin, B. 1993. English Verb Classes and Alternations: A Preliminary Investigation. University of Chicago Press, Chicago, USA.

Address of the author: Oxford University Press
Coach House South
Malling Deanery
Lewes
East Sussex BN7 2JA
UK

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 73-94 (1992-93)

DICTIONARY ENTRY PARSING BASED UPON METHODICAL SEGMENTATION AS A PREREQUISITE FOR A PROJECTED META-LEXICOGRAPHIC WORKBENCH

CHRISTOPH BLÄSI – HEINZ-DETLEV KOCH

1. Introduction

Dictionary entries have an implicit textual structure which is intuitively made of use by every dictionary user. There have been various attempts to make those structures explicit and to describe them formally (cf. Wiegand 1991). Since such descriptions are meanwhile available in a relatively explicit and formal manner, it is worthwhile to examine if it is possible to develop algorithms for parsing such structures. In case it is one can expect that techniques can be applied that have been developed and tested in other academic disciplines with comparable object domains.

Motivated by these considerations we have tried to refine those descriptions to such a degree that an operationalization is possible and that their correctness can be shown using a dictionary entry parser. For three reasons, the parsing of dictionary entries has become an important field within computational lexicography :

- The specific structure of dictionary entries has proven to be suitable for the representation of knowledge of language and the world. Since centuries, this knowledge has been organized around lexemes and this structure is therefore an interesting object of research.
- Since the extent of dictionaries as well as demands for their quality have increased, machine support for lexicographers is desirable also with regard to structural aspects of dictionary entries.
- From the side of natural language processing the demand for this kind of knowledge has risen to a degree which, at least for broad coverage applications, cannot entirely be satisfied by new coding.

For the qualified extraction of structural knowledge the parsing of dictionary entry structures is an indispensable prerequisite (cf. McNaught-Caroli-Hellwig 1990).

Additional desirable features of such dictionary entry parsing systems are that they should be as general as possible (i.e. they should be applicable to

Akadémiai Kiadó, Budapest

dictionaries of most different kinds without substantial changes) and that they should fall back upon components of conceptionally similar applications as far as possible.

In the following, we will present a system which has been developed at the University of Heidelberg between 1990 and 1993 and its lexicographic and computational rudiments. We will demonstrate its functionality using examples from the DDUW (Duden Deutsches Universalwörterbuch). The system arose from several of the listed motivations for the parsing of dictionary entries at the same time; it is especially designed to meet the requirements for generality in the sense given above and makes extensive use of already existing components.

2. Rudiments

The "meaning" of a string in a dictionary entry is determined by this string itself and its location within the dictionary entry, i.e. its position within the specific structure of the latter. The parsing of a dictionary entry can be carried out only with respect to a certain concept of structure. The choice for the constituent structure paradigm seems to be obvious in the case of dictionary entries. Immediate dominance and linear precedence relations as occurring in those entries are usually represented by corresponding constituent structure trees. With the help of the method of "exhaustive positional-functional segmentation" (Wiegand 1989a, 1989b) of dictionary entries such constituent structure trees can be derived. This derivation is first carried through for actual dictionary entries (see Fig. 1, left). However, abstractions can be made over whole classes of isomorphic constituent structure trees just by not considering the terminal elements (see Fig. 1, right).

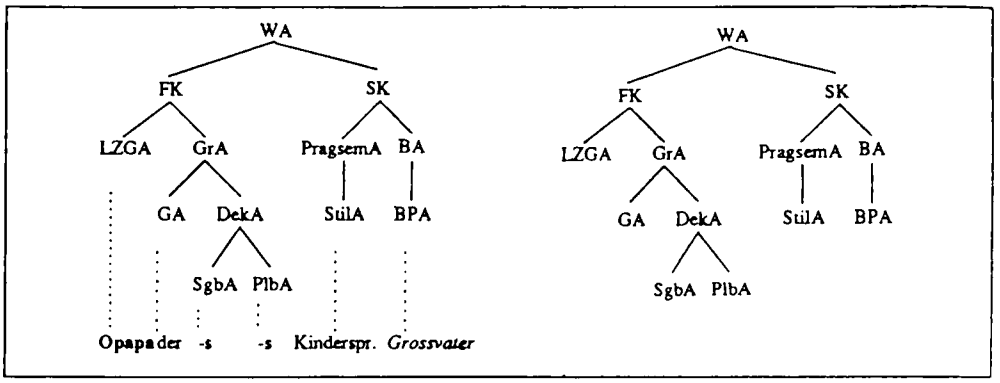


Fig. 1

For reasons of simplicity, neither typographic (e.g. font changes) nor non-typographic (e.g. comma, semicolon, colon, ...) structure indicators have been given. Abbreviations used: WA: Wörterbuchartikel / Dictionary Entry, FK: Formkommentar / Comment on Form, LZGA: Lemmazeichengestaltangabe / Item giving the Form of the Lemma Sign, GrA: Grammatikangabe / Grammar Item, GA: Genusangabe / Item giving the Gender, DekA: Deklinationsangabe / Item giving the Declension, SgbA: Singularbildungsangabe / Item giving the Singular Formation, PlbA: Pluralbildungsangabe / Item giving the Plural Formation, SK: Semantischer Kommentar / Comment on Semantics, PragsemA: Pragmatisch-semantische Angabe / Pragmatic-Semantic Item, StilA: Stilangabe / Style Item, BA: Bedeutungsangabe / Item giving the Meaning, BPA: Bedeutungsparaphrasenangabe / Item giving the Meaning Paraphrase

For a linear representation of such tree structures production rules of the form A → B C or list structures (A (B C)) are suitable. In the example given, A consists of B and C or B and C are parts of A, respectively. The immediate dominance and linear precedence relations stated by these rules are the by far dominating ones, but not—as hinted at above—the only relations occurring in dictionary entries; for the representation of further relations (e.g. the addressing relation and the scope relation, which are both independent from the constituent structure) a representation as tree structure or production rule does not suffice. For this, one needs corresponding annotations the details of which are beyond the scope of this paper (cf. Bläsi 1991a, 1991b, 1991c). Actual dictionary entries can be attributed structure by mapping them to the abstractions of the corresponding class of dictionary entries. The rules generating the set of all such abstract constituent structure trees are taken as grammar. A slightly simplified extract from the grammar for noun entries of the DDUW is given below:

WA → FK %EKA EtyA %EKZ SK %PKT
WA → FK SK
FK → bold LZGA %KOM normal GrA
GrA → GA %SEM DekA
GA → TEXT
DekA → SgbA %KOM PlbA

%EKA: Angular bracket, opening; %EKZ: Angular bracket, closing; %PKT: Period; %KOM: Comma; %SEM: Semicolon

3. The analysis technique

The structural analysis is carried out relative to an extended version of the phrase structure grammars introduced in section **2**. Techniques for the analysis of strings based on these context-free grammars have been examined and compared for formal as well as for natural languages (Aho 1987; Hellwig 1990; Tremblay 1985). They are applied to a large extent in the various disciplines of computer science as well as in computational linguistics. For our objective we have chosen a special technique geared to the analysis of formal languages. Below, we will describe this technique and the modifications made for the processing of natural language and, in an additional step, dictionary entries.

The chosen technique has originally been developed for a subset of formal languages. This set of languages can be described by certain context-free grammars which are called LR(1) grammars. An operational definition of this class of languages is based on the definition of an abstract, deterministic, finite automaton, which is seen as a representative of a LR(1) grammar. All languages which can be generated by such an automaton belong to the class of the so-called LR(1) languages which also includes most of the programming languages used today. LR(1) languages have the advantage that the abstract automaton which generates such a language can likewise be applied to its analysis. Fortunately, this analysis can be performed in linear time.

Genuine LR(1) languages are built on atomic categories and do not allow for ambiguous grammars. When analysing an input string, the automaton steps through a finite set of states and, triggered by its current state and the current input symbol, decides whether to push the current input symbol on a stack (Shift) or to replace some of the symbols already on the stack by a nonterminal symbol (Reduce). By performing any of these actions, the automaton changes its state according to the symbol on top of the stack and the current input symbol. A grammar is an LR(1) grammar if one could construct

an automaton that contains only one possible action for every pair of a state and a symbol.

LR(1) grammars were modified for natural language processing by Tomita (1985a, 1985b) who introduced a graph-structured stack to allow for multiple alternative actions for any state-symbol combination. If the automaton encounters a state with more than one possible action the algorithm forks the stack and pursues the resulting alternatives in parallel. Analysis of an input string succeeds if at least one of the alternatives succeeds.

We have further modified the Tomita approach by allowing for complex, i.e. feature augmented, categories which contain pairs of features and feature values. By imposing restrictions on the possible feature values, we can bidirectionally enforce identity of values by mutual instantiation of variables or restrict values to sets of admissible values for a feature. By restricting unification processes to local trees, i.e. trees of depth 1, we are avoiding the time-consuming complexity of full graph unification and are thus combining the speed of LR(1) analysis with the expressive power of general context-free grammars.

As input for the parsing process we are using the machine-readable representation of the dictionary, i.e. the computer typesetting tape.

4. The processing of the typesetting tape

A typesetting tape is the result of a computing operation geared to a specific typesetting machine which processes the input text according to the typographic requests on the part of the typesetter. Among other tasks, this operation determines the distribution of the textual material to lines, colums, pages and sheets. As the result of this step, a typesetting tape contains the informational content as well as informations concerning the typography and layout of the text. The latter informations are coded in a language specific to the typesetting machine used. Some of them, as for example the ones concerning the page makeup, are totally irrelevant for determining the structure of a dictionary entry. Others, however, especially font changes, provide indispensable information on the organisation of the dictionary entry. Without the latter an entry is hardly comprehensible even to the human reader of the printed version.

Whereas irrelevant control sequences as exactly e.g. the ones concerning the page makeup or additional blanks after italicised text passages can thus safely be ignored, other information carrying control sequences have to be recognized. They can, moreover, be replaced by less machine dependent and

more generally understandable characters. Before the processing, an extract of the type setting tape of the DDUW looks like this:

```
ð5Qÿ-Ona+ger, ð1Qder; -s,---||lat. onager, onagrus }} }||}
griech. 'onagros; 2: nach der einem Esel glei}-}
chenden Form||: ð3Q1. ð2Qin Südwestasien heimi}-}
scher Halbesel.ðVR1QQ ð3Q2. ð2Q (im antiken Rom) Wurfma}-}
schine.æ
```

This typesetting tape produces the following print output:

Ona|ger, der: -s, - [lat. onager, onagrus < griech, ónagros; 2; nach der einem Esel gleichenden Form]: **1.** *in Südwestasien heimischer Halbesel.* **2.** *(im antiken Rom) Wurfmaschine.*

Fig. 2

In a first step of the processing of the typesetting tape, control characters specific to the typesetting system are replaced by the ones chosen for the analysis process, structurally irrelevant sequences are deleted. After this step the same extract of a typesetting tape looks like this:

```
%FONT5 %USTR1 Ona %VSTR ger %KOM %FONT1 der %SEM %HSTR s %KOM
%HSTR %EKA lat %PKT onager %KOM onagrus %LT griech %PKT %ALR
onagros %SEM 2 %DPP nach der einem Esel gleichenden Form %EKZ
%DPP %FONT3 1 %PKT %FONT2 in Südwestasien heimischer Halbesel
%PKT %FONT3 2 %PKT %FONT2 %KLA im antiken Rom %KLZ Wurfmaschine
%PKT
```

The replacements mentioned above are of a purely textual nature and can be carried through with any text editor.

Owing to the multifunctionality of the remaining delimiters the "textual" units cannot yet be identified as string units of an entry in a straightforward way. Those delimiters have a structuring function as punctuation marks on the linguistic level on the one hand and explicate the organisation of the dictionary entry as condensed text on the other. This class of structure indicating information cannot be separated from the informational content without some effort. Details will be given in section 5.

5. The segmentation problem

Since the system in question is a dictionary entry parser, its structural view is necessarily restricted to structures constituting the dictionary entry as the object of investigation. Every item (as a string at a certain position in the dictionary entry structure), however, has itself a (subordinated) structure (namely e.g. a sentence structure or a phrase structure—this can, of course, be continued to the morphological and phonological level). Those subordinated structural levels are not accessible to the view of a dictionary entry parser. The problem is that it is not clear from the outset which sentences and phrases, respectively, are units not to be pervaded further by the dictionary entry parser and which are not.

We have tried several strategies to solve this segmentation problem. Only one of those solutions, the most radical and least "intelligent" one, succeeded.

- All strings which appear between certain types of structure indicators (semicolon, colon, font change, ...) are interpreted as string units. From what has been said in the previous paragraph it is clear that this approach must fail. Some of the delimiters can mark the boundary of string units as well as occur within such units. An angular bracket in the DDUW can e.g. mark the boundary of an Item giving the Etymology (as string unit, from the Comment on Form and from the Comment on Semantics) as well as introduce the giving of facultative letters within a Grammatical Item.
- Everything which lies between certain boundaries is interpreted as a string unit. These boundaries have to be found—with the aid of the structure indicators already mentioned—by context-sensitive matching or with the help of a context-sensitive transition network. As for angular brackets, for example, it has to be decided, if they are situated within a Grammatical Item (in this case the angular bracket just occurs within a string unit) or not (the angular bracket marks the boundary between string units). This approach is inappropriate because much of the structural knowledge which is desired as the result of the actual parsing process is needed for the preprocessing already. For the decision on the angular bracket one needs to know if an Item giving the Form of the Lemma Sign has been consumed or not.
- A "maximal" segmentation is performed. All typographic structure indicators and all delimiters which in at least one of their usages can mark such boundaries are taken to be the set, with respect to which this segmentation is carried through. Strings which have been split "overeagerly" are

re-concatenated in a later stage of work—namely during parsing. This applies especially to the case that a delimiter has been interpreted as marking a boundary, whereas it functioned as a punctuation mark. For this approach, however, the grammar has to be slightly changed. It has to be allowed that items can be realized not only as units, but alternatively as chain of such units. If one makes sure that the corresponding rules are defined strictly right or strictly left recursively this strategy leads to very satisfying results. (The latter precaution is to prevent that the same overall string occurs several times because of various derivation possibilities from smaller component units; the runtime efficiency would be influenced in a very negative way.)

6. Description of the implementation

The grammar for the analysis is written in a formalism following the programming language Lisp and resembles the usual notation for production rules. A special preprocessing program compiles this grammar into a control table for the automaton. The output of the preprocessing step is loaded by the actual analysis program which contains a rudimentary LR(1) automaton. By using this control table the analyzer is capable of attributing structure to all input strings which are covered by the grammar.

An extract of the grammar for the DDUW which describes the arrangement of the Comments on Semantics can be represented in production rule notation as follows:

```
SK   →  %DPP SKK
SKK  →  PA SSK
SKK  →  PA SSK %PKT SKK
```

This fragment dwells on the fact that a Comment on Semantics consists of the categories %DPP (for the non-typographic structure indicator colon) and SKK (for the Complex of Comments on Semantics), whereas SKK itself consists of a PA (Item giving Polysemy) and a SKK (Subcomment on Semantics). The third rule determines that a Complex of Comments on Semantics can be followed by further Complexes of Comments on Semantics which are seperated by %PKT (for period). The formalism as used by the system differs from this notation syntactically by just combining all involved categories in a list the first element of which is the superordinate unit. The fragment shown above is rewritten as follows:

```
[1]  ( (SK) (%DPP) (SKK) )
[2]  ( (SKK) (PA) (SSK) (%PKT) (SKK) )
[3]  ( (SKK) (PA) (SSK) )
```

The necessity for additional bracketing of every category is obvious if we consider a rule which shows complex categories. In this case all specifications related to a category are combined in the one bracket, whereas the features and the feature values are themselves enclosed in brackets. Feature values which have to be instantiated identically are notated as bracketed variables. Variables with identical names have to be instantiated by identical values at runtime if they occur in the same production. The rule below describes the form of an Item giving Polysemy in the DDUW:

```
[4]  ( (PA (index (x))) (TYPO-ANF (art fett)) (TEXT (string (x))) (%PKT) )
```

The category PA consists of a typographic structure indicator TYPO-ANF, a text and a period. The index of the Item giving Polysemy is assigned the value of the feature `string` of text via a reoccurring variable `x`. The feature `art` of the typographic structure indicator is constrained to the value `fett`.

The preprocessing step produces a control table with two types of entries: actions to be carried through and target states. For completeness reasons one entry of each of these types is given below:

```
Actions:      ((%DPP (SHIFT 33)) (%PKT (REDUCE 28)))
Transitions:  ((SSK 57) (PRAGSEMA-SSK 56) (BA 55))
```

The program takes about 3.5 minutes to produce a control table with 224 states for a grammar with 117 rules. Since the control table is written to a file which is loaded by the actual analysis program, this step has to be performed only when the grammar has been changed.

The analysis program Paula (Parser for Ambiguous Unification grammars / Lr(1)-Analysis) is completely written in Lisp and contains essentially a rudimentary LR(1) automaton, a component for the lexicon search and a module for the graphic output of the results. Apart from the control table and the actual input, it requires a lexicon which establishes the connection between the units occurring in the input text and the categories used for the grammar. The units occurring in the lexicon are the ones in the processed typesetting tape. An extract of the lexicon used for the analysis of the DDUW has the following form:

```
(%KLA %KLA (string "(") )
(%KLZ %KLZ (string ")") )
(%SEM %SEM (string ";") )
(%PKT %PKT (string ".") )
(%DPP %DPP (STRING ":") )
(%KOM %KOM (string ",") )
(%SKA %SKA (string "<") )
(%SKZ %SKZ (string ">") )
(%FONT5 TYPO-ANF (ART fett))
(%FONT1 TYPO-ANF (ART normal))
(%FONT2 TYPO-ANF (ART kursiv))
```

Again, the notation is oriented towards Lisp and describes a lexicon entry as a list. The first element of the list is the symbol actually occurring in the processed typesetting tape, whereas the rest of the list contains the information to be provided by the lexicon component, as soon as this unit occurs in the input. For the analysis of dictionary entries, the lexicon component of the parser was modified so that it provides the category text if a string enclosed in double quotes occurs in the input. This string is attributed to the feature string of this category. Therefore all elements of the input which have been enclosed in double quotes by the typesetting tape processing program can be treated by the parser as string units. In rule [4] above the index of an Item giving Polysemy, i.e. the number in front of the period taken as text, is attributed to the feature string of the category text by the lexicon phase of the parser and thus propagated to the superordinate category PA.

Initially, the program loads the control table as well as the lexicon and starts with the analysis of the input file which may contain an arbitrary number of articles from the preprocessed type setting tape. After the analysis of an entry, its structure is usually displayed in a window which the user can move over the constituent structure tree. When choosing a category with the mouse, its accompanying features are displayed in an additional window. Figure 3 shows an extract of the structure of the DDUW entry ONAGER as displayed by the output component:

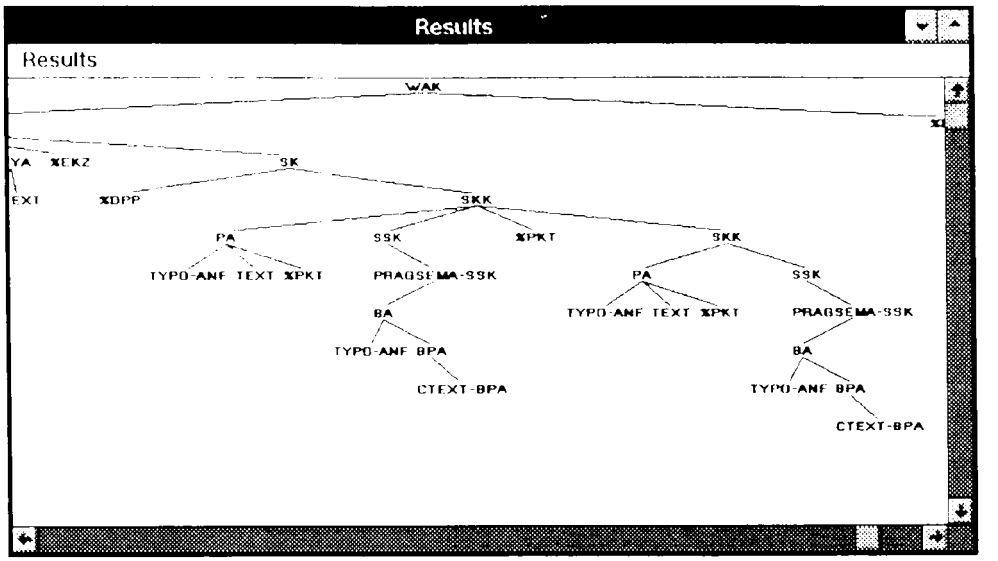


Fig. 3

Alternatively, the structures produced can be written to an output file in list notation. This output can be processed further by other programs. Such programs could be used to extract certain types of information, e.g. grammatical information, or to transfer the results to a lexicographic database (cf. Koch 1993). Optionally, all not analyzed inputs, i.e. all entries which were not covered by the grammar, can be written to a second output file to be postprocessed manually or to serve as guide for the extension of the grammar. Paula puts two tools for the grammar development at the disposal of the user. It is possible to run the analysis in single step mode. The program interrupts the analysis after each action and offers the possibility to examine the actual status of the automaton as well as to check the analysis steps carried out already. As a further option, Paula can interrupt the analysis if one of the running analysis processes terminates without a result in order to give the program user the possibility to investigate the reason for the failure. The usual proceeding for drawing up a grammar consists in developing a rather small grammar for a part of the entries to be examined which is applied nonetheless to the entire typesetting tape. Entries not analyzed are written to a file. By examining those not accepted it is usually possible to determine rather soon where and in which way the grammar has to be extended. In case of a lack of clarity concerning the reason for the failure, the mechanisms described above

can be applied to gain more detailed information. Using the modified grammar for the next analysis attempt the number of not accepted entries is supposed to decrease and allows for a renewed extension of the grammar.

The output component has to cater for the consequences of the segmentation strategy chosen (cf. section 5). The segmentation of the typesetting tape into units as small as possible results in inadequately hierarchical subtrees consisting of recursively descending pairs of delimiters and rest trees for the textual constituents. The subordination of these constituents under the same textual category, however, indicates the fact that during the processing of the typesetting tape a too finegrained segmentation has been assumed. Therefore the produced constituent structure tree is searched for the occurrence of textual constituents after the actual analysis process has finished. All constituents directly or indirectly subordinated to such a category are deleted and the values of their string feature are "collected" and concatenated. The string resulting from this is attributed to the feature compound-string of a new main node, replacing the whole subtree.

Technical details of the implementation

The program producing the table as well as the analysis program are totally independent on the dictionary to be examined. Paula takes about 1.5 seconds for the analysis of a dictionary entry of average size and complexity using the grammar applied at the moment. For the implementation we used Golden Common Lisp, Version 4.1 with the Gold Hill Windows extension, Version 4.1 under Microsoft Windows 3.1 on a IBM compatible PC with a 33 MHz 80386 processor which is equipped with 9 MB RAM and a SVGA graphics card with 800×600 pixels.

For the processing of the typesetting tape we used an ordinary text editor with macro-facility (KEdit 4.0) and a simple C program. These editor macros and the C program, however, are dependent of the typesetting system used and have to be adapted accordingly if the system is applied to a dictionary which is produced by a different typesetting system or read with an OCR scanner.

Independently from this fact, a grammar has to be drawn up for each dictionary to be examined.

7. Example

In the following, the structure produced for the DDUW entry ONAGER is shown. To improve readability, the lists have been indented according to their

structure and informations not relevant for the entry structure have been removed. The markings :T, :C, and :F occurring in the output indicate the type, the constituents, and the features of the different categories.

```
(:T WAK    :C
    ((:T WA    :C
       ((:T FK :F ((KLASSE SUBST)) :C
            ((:T TYPO-ANF :F ((ART FETT)))
              (:T LZGA    :C
                  ((:T CTEXT-LEM :F ((COMPOUND-STRING "--Ona!!ger")))))
            (:T %KOM :F ((STRING ",")))
            (:T TYPO-ANF :F ((ART NORMAL)))
            (:T GRA    :C
                ((:T GA :C
                   ((:T TEXT :F ((STRING "der")))))
             (:T %SEM :F ((STRING ";")))
             (:T DEKA :C
               ((:T SGBA :C
                    ((:T %HSTR :F ((STRING "-")))
                     (:T TEXT    :F ((STRING "s")))))
                (:T %KOM :F ((STRING ",")))
                (:T PLBA    :C
                    ((:T %HSTR :F ((STRING "-")))))))))))
        (:T %EKA :F ((STRING "[")))
        (:T ETYA    :C
            ((:T CTEXT :F ((COMPOUND-STRING "lat.onager,onagrus<griech.
                    'onagros;2:nach der einem Esel
                    gleichenden Form")))))
        (:T %EKZ :F ((STRING "]")))
        (:T SK    :C
            ((:T %DPP :F ((STRING ":")))
              (:T SKK    :C
                  ((:T PA :F ((NUMMER    "1")) :C
                     ((:T TYPO-ANF :F ((ART FETT) (SUBTYP ?)))
                       (:T TEXT :F ((STRING "1")))
                       (:T %PKT :F ((STRING "."))))))
                   (:T SSK    :C
                     ((:T PRAGSEMA-SSK    :C
                          ((:T BA    :C
```

```
    ((:T TYPO-ANF :F ((ART KURSIV)))
     (:T BPA :C
       ((:T CTEXT-BPA :F ((COMPOUND-STRING
"in Südwestasien heimischer Halbesel")))))))))))
(:T %PKT :F ((STRING ".")))
(:T SKK    :C
  ((:T PA :F ((NUMMER "2")) :C
       ((:T TYPO-ANF :F ((ART FETT) (SUBTYP ?)))
        (:T TEXT :F ((STRING "2")))
        (:T %PKT :F ((STRING ".")))))
   (:T SSK    :C
       ((:T PRAGSEMA-SSK :C
          ((:T BA :C
             ((:T TYPO-ANF :F ((ART KURSIV)))
              (:T BPA :C
                  ((:T CTEXT-BPA :F ((COMPOUND-STRING
"(im antiken Rom)Wurfmaschine")))))))))))))))))))
(:T %PKT :F ((STRING ".")))))))
```

8. Related work

There have been several approaches to dictionary entry parsing. The approach followed by IBM (cf. e.g. Neff–Boguraev 1989) is probably the most widely known among them. Continental applications of the underlying ideas have been described for example by Wermke–Bläser (1990) and by Marinai–Peters–Picchi (1990).

As far as we can see, these approaches have analyzed a wide range of interesting and complex phenomena in dictionary entries. Moreover, they have reached an astonishing high degree of parsing coverage. However, we have shifted emphasis in at least two dimensions:

- The rules used in our system are as it were natural by-products of actions in the established lexicographic theory on dictionary microstructures (Wiegand 1989a, 1989b). The writing of a corresponding dictionary entry grammar falls directly back to the results of the afore-mentioned "exhaustive positional-fuctional segmentation" (and, we would like to add, classification) in this theory. This segmentation method deserves the name method in so far as all segmentation steps are described algorithmically. Moreover, a clear terminology for these well-defined segments has been introduced which further eases the handling of the rules.

As opposed to all this, writing rules in the IBM paradigm encourages ad-hoc-theorising on the nature of dictionary entry structures and ad-hoc-naming of certain parts of them.

- We consider dictionary entries as texts with a characteristic text structure which therefore should be made explicit without destroying its effects on the appearance in printed form. Dictionary entries are the result of a condensation process from running text describing linguistic entities. It is idiosyncratic to single dictionaries, constitutes their styles to a large degree and cannot easily be formulated as algorithms. Using attribute-value-pairs for the indication of "cross-tree" relations (e.g. scope and addressing as mentioned above), we can make all relations, including these cross-tree ones, explicit while preserving the structure of the dictionary entry structure as a condensed text structure.

  Whereas the strategy "to represent precisely all information contained in the printed entry—and, wherever possible, to represent explicitly information which is only given implicitly" (Marinai–Peters–Picchi 1990) means duplicating subtrees with information referring to e.g. two other subtrees in a non-trivial way, it is our strategy to uniquely mark this subtree as referring to the two others, i.e. as having those two in its scope.

9. Applications

As far as the lexicographical process, i.e. the process of producing dictionaries from setting up an appropriate project plan up to the delivery of the printing tape, is concerned, we can imagine two groups of applications of dictionary entry parsing.

9.1. Consistency check

One of the main advantages of an efficient dictionary entry parser is the possibility to perform with its help a structural concistency check with respect to a given grammar during various production stages of a dictionary. Two possible scenarios shall be described as follows:

a) While the dictionary entries are written, each single entry can be checked to verify that this entry satisfies the requirements stated in the grammar. The only prerequisite is an arbitrary, but consistent tagging formalism, possibly as simple as one provided by the formatting capabilities of a text processor.

b) Once the writing of the dictionary has been completed, the result can be run through the parser. During the parsing process, all entries that do

not adhere to the requirements stated in the grammar are rejected and form a list of structurally erroneous dictionary entries. Furthermore, the resulting list of correctly analysed entries can be subject to further checks. Since the inherent structure of each dictionary entry has been made explicit, it is easy to scrutinize all cross-references and to verify that they are neither cyclic nor void.

We are currently working on two extensions to the Paula-System, based on its MS-Windows implementation:

1) Data Communication: A DDE interface is being added that incorporates a configurable scanning routine that resembles the functionality of the external preprocessing programs described above. Using this approach, we hope that we will be able to establish an interaction of Paula and an arbitrary text processor, e.g. WinWord. By selecting a portion of text that corresponds to a single dictionary entry and passing the selected text to Paula, a consistency check of a single entry can be carried out that will constitue a considerable subset of the functionality of markup-oriented editors without the investment and need for additional training usually associated with such editors.

2) Functionality: Closely related to this approach is an extension of Paula to provide more explanatory error messages in the case of parse failure. Two approaches are being followed:

a) The first approach employs the error recovery method developed by Penello and deRemer (cf. Tremblay 1985). This method works by succesively applying different repair strategies on the parsing configuration when the parser encounters an error and selecting the locally least expensive repair action (where the costs are measured in terms of the number of operations involved in re-establishing a legal configuration and the (weighted) costs of the operations themselves).

b) The other approach is inspired by considerations concerning checking and correcting natural language. Because with dictionary entry parsing the reasons for parse failure usually do not lie in lacking feature agreement—as is typically the case with natural language—the most advanced techniques for error detection and correction in natural language, unification failure approaches, are neither suitable nor necessary for dictionary entry parsing. Every other technique, though, basically relies on rule relaxation (applying less rigid rules corresponding to the ones that fail(ed)).

Rule relaxation, however, implies predicting possible errors; whereas this is certainly too much of a reductionistic approach with natural language, it can be considered sufficiently appropriate for the restricted "language" of dictionary entry structures. This applies to predictable errors in constituent se-

quence as well as with unpaired structure indicators. Core rules must in pairs be connected to peripheral rules (rules generating "wrong" structure), in case parsing using only core rules fails.

Repair actions and peripheral rules trigger error messages dwelling on the nature of the structural deficiency of the structure connected to the error or only parsed by peripheral rules, respectively. The problem of the latter approach is that errors do not only have to be predicted as mentioned earlier, but that, moreover, every single appropriate error message has to be formulated by the writer of the dictionary entry grammar.

With both approaches we will be able to provide more detailed messages than "You can't do this now" usually found in markup oriented editors; we anticipate our system to generate error messages like "Dash expected as beginning of an item giving the singular formation; found instead: angular bracket".

9.2. Tagging for further processing

The most discussed application of dictionary entry parsers is their use use to make dictionary data as a result of traditional editorial processes (usually in the form of typesetting tapes) accessible to up-to-date editing environments, mostly built around lexical databases. In fact, this was the original aim of the approach. To achieve this goal, the representation of the structure in the parser output has to be translated into the one in the respective target formalism; this translation is an isomorphic mapping and can be done with the help of simple converting programs. The traditional setting is shown in a), whereas a more general approach is dwelled upon in b).

a) The target formalisms can be the proprietary taggings of the single subsystems within the editing environment, e.g. a certain typesetting system or a certain platform for electronic products. The corresponding converting programs have to be taylor-made for the subsystems.

b) The target formalism can be implementation independent tagging according to the SGML standard (ISO 1986; Goldfarb 1990). This SGML tagging can be used for almost all editing house purposes via standard interface programs, e.g. ones that load SGML tagged input into a relational database, preserving its structure. Further processing can then be carried through using standard tools. Corresponding possibilities have been described abundantly, it suffices to mention the online update of dictionary contents using an SGML editor. Possible applications using implementation independent tagging include the reformatting of the output, the extraction of selected information items, the merge of information items from different dictionaries, and the construction of data views of the dictionaries for lexicographers, linguists and others.

Following this, the parser output can easily be translated into a tagged representation:

```
<entry>
    <lemma> Onager </lemma>
    <article> der </article>
    <singular>---</singular>
    <plural> -s </plural>
    <part-of-speech> subst </part-of-speeech>
     . . .
</entry>
```

Using simple "resolutions" as described in section 10, this structure can be transformed into the following, even more self-explanatory structure:

```
<entry>
    <lemma> Onager </lemma>
    <gender> masculine </gender>
    <singular-form> Onager </singular-form>
    <plural-form> Onagers </plural-form>
    <inflection-class> 6 </inflection-class>
    <inflection-type> regular </inflection-type>
    <part-of-speech> noun </part-of-speeech>
     . . .
</entry>
```

10. Further perspectives

10.1. "Semantics" of dictionary entry structures

As for the second and the third of the possible motivations for dictionary entry parsing in computational lexicography given in the introduction, this system certainly proves the feasibility of the approach described above. By developing a running dictionary entry parsing system, the information contained in printed dictionaries (or the accompanying typesetting tapes, respectively) is by far not exploited and processed exhaustively. The following steps are conceivable and desirable for the future:

- The informational content of the units not to be considered with regard to their internal structure in the present context, for example the Items on Meaning, should be treated as such and represented in a suitable manner. This has e.g. been done in the ACQUILEX project (cf. Calzolari–Zampolli 1989).
- Since there is at most an arbitrary and traditional relation between the questions of a potential dictionary user and the place in the dictionary entry in which this particular question is answered, different syntactic configurations of the same "answer" across different dictionaries should be given identical semantics. Therefore, dictionary structures as wholes should be examined with respect to their semantics. Even if dictionary entries cannot be interpreted as functions or other well-researched mathematical objects, there is always the possibility for translation semantics which just defines the semantics of the "language" in question in terms of another "language" or formalism which has been given semantics already. Representatives of such semantically well-defined formalisms—some of which in addition have the advantage of being familiar to most of the lexicography community—are PROLOG, DATR (cf. Gazdar–Evans 1990; Gibbon–Ahoua 1991) and Typed Feature Structures (cf. Ait-Kaci 1986). Owing to the nature of lexical knowledge, default and inheritance mechanisms are needed for reasons of generalisation.

The fact that OPAPA has an "s" as its plural ending (cf. section **2**) is "coded" along the path `LZGA – FK – GrA – DekA – PlbA` in the DDUW. A desirable translation into (Pseudo-) PROLOG would be

```
Has_Plural_Ending (Opapa, s)
```

one in DATR

```
Opapa:  <FK LZGA>            ==  Opapa
        <FK GrA DekA PlbA>   ==  s
        <>                   ==  Noun_rule.
```

Necessarily, the predicates (in the PROLOG case) and path elements (in the DATR case) should be constrained to a defined repertoire oriented to a taxonomic level of linguistic description (Hellwig–Minkwitz–Koch 1990; cf. Heid–McNaught 1991) and/or potential dictionary user questions (corresponding studies are a desideratum).

With the help of "resolutions" of the type

```
Has_Plural_Ending (X, s) → Has_Plural (X, X^s)
```

("^" is the concatenation operator) or

```
Noun_Rule: <Plur>=="<FK LZGA>" ^ "<FK GrA DekA PlbA>".
```

("^" as above) respectively, lexical generalisations can be tackled appropriately. Technically, such translations (yielding semantics implicitly via the afore-mentioned translation semantics) can rather easily be achieved during parsing by means of a syntax oriented translation using attributed grammars (Knuth 1968; cf. Deransart–Jourdan–Lorho 1988). The translation results are dictionary-independent and it is thus possible to load different source dictionaries with different dictionary entry structures into a common lexical database: additional information can be ignored if stored already and added if not. In the case of conflicting information, strategies as for example "Dictionary A is usually more reliable than dictionary B" can be applied. For both steps a running, theoretically sound and well understood dictionary entry parsing system is necessary.

10.2. Steps towards a meta-lexicographic workbench

Based upon Wiegand's notion of meta-lexicography as theorizing on the lexicographic process, a meta-lexicographer concerned with structural aspects should at least be given assistance in drawing up—usually rather complex—dictionary entry grammars as used above. Such assistance should consist of a rule editor including syntactic help; this rule editor should allow for a more powerful and intuitive grammar formalism, a component for the visualization of rule interdependencies and complex structures. A transformation component should be reponsible for translating this user-oriented formalism into the one internally used by Paula.

Although this is a prerequisite, a full-scale meta-lexicographic workbench must comprise a considerable amount of additional tools; among them we could imagine presentation systems based on rapid prototyping or a statistics package.

As long as the focus of interest in meta-lexicography rests on structural features of dictionaries, a tool as the one described in this paper will prove to be an essential building block of meta-lexicographic (and lexicographic) research and "production" environments.

References

Aho, A.–Sethi, R.–Ullman, J. 1987. Compilers.

Ait-Kaci, H. 1986. An algebraic semantics approach to the effective resolution of type equations. In: Theoretical Computer Science 45: 293–351.

Bläser, B.–Wermke, M. 1990. Projekt "Elektronische Wörterbücher/Lexika": Abschlußbericht der Definitionsphase, IWBS [Institut für wissensbasierte Systeme der IBM Deutschland] Report 145. Heidelberg.

Bläsi, C. 1991a. Einige Überlegungen zum WA-Parsing. Internal Paper. Heidelberg.

Bläsi, C. 1991b. Einige weitere Überlegungen zum WA-Parsing. Internal Paper. Heidelberg.

Bläsi, C. 1991c. A forthcoming system for the "reusable" parsing of dictionary entries. Internal Paper. Pisa.

Bläsi, C.–Koch, H.-D. 1991 (1992). Maschinelle Strukturerschließung von Wörterbuchartikeln mit Standardmethoden. In: Lexicographica 7.

Calzolari, N.–Zampolli, A. 1989. Lexical databases and textual corpora: a trend of convergence between computational linguistics and literary and linguistic computing. In: Proceedings of the ALLC/ACH Conference in Toronto 1989.

Deransart, P.–Jourdan, M.–Lorho, B. 1989. Attribute Grammars. Heidelberg.

Drosdowski, G. *et al.* (eds). 1989. Duden Deutsches Universalwörterbuch.[2] Mannheim–Wien–Zürich.

Evans, R.–Gazdar, G. (eds). 1990. The DATR papers. Cognitive science research paper 139. University of Sussex.

Gibbon, D.–Ahoua, F. 1991. DDATR: un logiciel de traitement d'héritage par default pour la modélisation lexicale, English/Linguistics Interim Report No. 4. Bielefeld.

Goldfarb, C.F. 1990. The SGML Handbook. Cambridge.

Heid, U.–McNaught, J. 1991. Eurotra-7 Study: Final Report. Bruxelles–Stuttgart.

Hellwig, P. 1990a. 31—Parsing natürlicher Sprachen: Grundlagen. In: Computational Linguistics Computerlinguistik (HSK 4). Berlin–New York.

Hellwig, P. 1990b. 32—Parsing natürlicher Sprachen: Realisierungen. In: Computational Linguistics Computerlinguistik (HSK 4). Berlin–New York.

Hellwig, P.–Minkwitz, T.–Koch, H.-D. 1991. Eurotra-7 Study: DOC-9. Standards for Syntactic Description. Bruxelles.

ISO 8878. 1986. International Organization for Standardization: Information Processing.–Standard General Markup Language (SGML). Geneva.

Knuth, D.E. 1968. Semantics of context-free languages. In: Mathematical Systems Theory 2/2, 127–45. Correction: Mathematical Systems Theory 2/5, 95–6.

Koch, H.-D. 1993. Eine Entwicklungsumgebung für lexikalische Datenbanken auf Basis klassifizierter Merkmalsstrukturen. Internal Paper. Heidelberg.

Marinai, E.–Peters, C.–Picchi, E. 1990. The Pisa Multi-Lexical Database System. An integrated system for the acquisition, maintenance and interrogation of mono- and bilingual LDBs, ACQUILEX, ESPRIT BASIC RESEARCH ACTION No. 3030, Twelve Month Deliverable. Pisa.

McNaught, J.–Caroli, F.–Hellwig, P. 1990. Eurotra-7 Study: DOC-1. Possible Applications of Reusable Lexical Resources. Bruxelles.

Neff, M. – Boguraev, B. 1989. Dictionaries, dictionary grammars and dictionary entry parsing. In: 27th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Vancouver.

Tomita, M. 1985. An efficient context-free parsing algorithm for natural language. In: Proceedings of IJCAI.

Tomita, M. 1985. Efficient Parsing for Natural Language. Dordrecht.

Tremblay, J.-P. – Sorenson, P. 1985. The Theory and Practice of Compiler Writing. New York.

Wiegand, H.E. 1989a. 38a—Der Begriff der Mikrostruktur: Geschichte, Probleme, Perspektiven. In: Wörterbücher Dictionaries Dictionnaires (HSK 5.1). Berlin – New York.

Wiegand, H. E. 1989b. 39—Arten von Mikrostrukturen im allgemeinen einsprachigen Wörterbuch. In: Wörterbücher Dictionaries Dictionnaires (HSK 5.1). Berlin – New York.

Wiegand, H.E. 1990 (1991). Printed dictionaries and their parts as text. In: Lexicographica 6: 1 – 126.

Addresses of the authors: Christoph Bläsi
Bibliographisches Institut & F.A. Brockhaus AG
Dudenstrasse 6
Postfach 10 03 11
D-6800 Mannheim
Germany

Heinz-Detlev Koch
Lehrstuhl für Computerlinguistik
Universität Heidelberg
Karlstrasse 2
D-6900 Heidelberg
Germany

Acta Linguistica Hungarica, Vol. 41 (1–4), pp. 95–108 (1992–93)

ANALYSE DES COMPOSES NOMINAUX DE L'ALLEMAND SUR LA BASE DES PROPRIETES SEMANTICO-SYNTAXIQUES DE LEURS CONSTITUANTS

DANIEL BRESSON

1. La notion de composés non lexicalisés

1.1. La notion de «Kompositum» ou de «Zusammensetzung»

Cette notion recouvre en allemand plusieurs acceptions et ne coïncide pas avec la notion française de «nom composé»(Bresson 1991; Gross 1988). L'ouvrage monumental sur les composés nominaux publié par l'Institut für Deutsche Sprache (Deutsche Wortbildung 1991, par la suite: SK) distingue de «vrais composés» («echte» Komposita), dans lesquels les unités lexicales conservent tout ou partie de leur sens propre: *Vereinsmitglied* ou *Landeplatz*, et des «nominalisations complexes» («komplexe Nominalisierungen»), dans lesquelles il y a entre l'élément A et l'élément B une relation «verbale» (SK 641–3), comme dans *Rheumabekämpfung*. Il s'agit, pour SK, d'un cas particulier de dérivation, et non de composition à proprement parler. C'est dans ce cas la base de type verbal B qui ouvre un certain nombre de possibilités de combinaisons syntaxiques qui seront actualisées par l'élément A (sujet, objet, agent etc . . .), comme cela a été montré par ailleurs (Kürschner 1974; Bresson 1991).

Une telle distinction entre des composés véritables et des composés-dérivés, parfaitement justifiée quand il s'agit de décrire le type de relation susceptible d'être réalisé entre A et B, ne l'est plus dans la perspective du traitement automatique. En effet, le seul critère immédiatement perceptible dans ce cas est celui de la polylexicalité, c'est-à-dire la présence dans des textes de mots complexes constitués par des chaînes de caractères non séparées par un blanc et comportant plusieurs unités lexicales identifiables. Si ces chaînes ne sont pas «lexicalisées», c'est-à-dire répertoriées dans des dictionnaires généraux ou spécialisés, des lexiques, glossaires ou autres ouvrages techniques, il est impossible de les analyser. Or ce phénomène est particulièrement fréquent en allemand, et l'on peut observer comment se font et se défont dans un texte des unités polylexicales éphémères mais parfaitement

Akadémiai Kiadó, Budapest

analysables en fonction du contexte et de la connaissance que les locuteurs ont des propriétés sémantiques, pragmatiques et syntaxiques de leurs constituants.

1.2. L'exemple de *Abfall*

Le dictionnaire Duden en un volume (Duden 1989), qui compte 120 000 items, mentionne sous l'entrée *Abfall* 'déchet' trois composés avec *Abfall* comme premier composant: *-eimer* 'poubelle', *-produkt* 'déchets, résidus' et *-rohr* 'sorte de gouttière'. Le Duden en six volumes (Duden 1977), 500 000 items, mentionne en plus de ces trois composés 10 termes supplémentaires: *-beseitigung* 'élimination des déchets', *-erzeugnis* 'production de', *-grube* 'fosse à ordures', *-haufen* 'tas d'ordures', *-korb*, *-kübel* et *-tonne* 'poubelle', *-material* et *-stoff* 'résidus', *-verwertung* 'utilisation des déchets'. Un dictionnaire spécialisé dans ce domaine (Seidel 1988) mentionne 41 composés possédant *Abfall* comme premier élément. Il s'agit essentiellement de termes spécialisés relevant des techniques de protection de l'environnement, et certains mots figurant dans les deux Duden n'y figurent d'ailleurs pas, comme *-eimer* et *-rohr*, car ils ne relèvent pas de cette technique.

D'autre part, une liste de noms établie à partir du corpus de l'IDS[1] fait apparaître 15 noms comprenant *Abfall* comme premier élément, mais seul dans cette liste *Abfallverwertung* est mentionné dans Duden ou Seidel. Voici ces composés sur *Abfall* rencontrés dans des textes: *-entsorgung* 'élimination des déchets- surtout radioactifs', *-entsorger* 'personne ou installation pour . . . ', *-vermeidung* 'éviter les', *-definition* 'définition des', *-konzept* 'notion de', *-satzung* 'réglementation sur', *-misere* 'grave situation des', *-beamter* 'fonctionnaire spécialisé dans', *-experte* 'expert en', *-management* 'gestion des', *-händler* 'commerçant en', *-rechtler* 'spécialiste du droit de'.

On constate qu'apparaissent dans les textes des composés nominaux qui ne sont pour la plupart d'entre eux ni des composés répertoriés dans les dictionnaires généraux, ni des termes relevant du lexique de la spécialité, le corpus de l'IDS ne comprenant pas d'ouvrages techniques à proprement parler. Ces composés nominaux ont pour second composant des noms abstrait comme *Definition*, *Management*, *Konzept*, ou des noms d'agent tels *Experte*, *Beamter*, *Händler*. A première vue, ces composés ne relèvent pas du mode de formation de termes spécialisés tels que *Abfallbelebtschlamm* 'boue active' ou *Abfalleinäscherung* 'incinération des déchets', qui désignent des techniques

[1] La liste à laquelle il est fait allusion ici est une liste alphabétique électronique de près de 200 000 noms établie à partir du corpus de l'Institut für Deutsche Sprache de Mannheim et qui a été très généreusement mise à ma disposition par cet institut. Pour tous renseignements concernant ce corpus, cf. Brückner (1986).

particulières et figurent dans le dictionnaire spécialisé. Les premiers appartiennent au contraire au domaine général et sont formés selon des paradigmes très courants: *A-definition*, *A-konzept*, *A-experte*. Les composés des domaines spécialisés doivent être répertoriés dans des dictionnaires et traduits dans la langue étrangère en tant qu'unités correspondant à un désigné identifiable. Il ne saurait être question d'essayer de les traiter comme des constructions libres et de leur appliquer les algorithmes que l'on utilise pour de telles constructions. Leur traitement relève du domaine de la terminologie plus que de l'analyse linguistique proprement dite. Mais on comprend inversement qu'il ne serait ni utile, ni facile d'essayer de faire des listes interminables d'unités lexicales ayant comme second composant des termes abstraits tels que *Definition*, *Konzept* ou *Experte*, qui peuvent être associés à un très grand nombre de mots en position A. Il me semble que cette constatation permet de bien situer et évaluer les problèmes posés par les composés lors du traitement automatique d'une langue comme l'allemand, qui offre à ses usagers la liberté de pouvoir concaténer en une seule unité graphique plusieurs unités lexicales sans former nécessairement pour autant une nouvelle unité lexicale. Cette liberté est largement utilisée par les locuteurs de l'allemand, et il faut donc s'attaquer au problème autrement, en tentant de d'analyser les relations possibles entre A et B.[2]

2. L'analyse de la relation entre les constituants A et B sur la base des propriétés de A et de B

Je vais prendre deux exemples de bases B désignant un concret: *Zeitschrift* 'revue' et *Tisch* 'table' et deux exemples de bases désignant un abstrait: *Methode* 'méthode' et *Technik* 'technique'. Il s'agit à la fois de noms figurant dans des dictionnaires, de noms rencontrés dans des textes et de noms figurant dans la liste de l'IDS, sans que le critère de l'attestation dans un dictionnaire puisse être considéré comme déterminant pour le statut lexical de ces mots complexes.

2.1. L'exemple de *A-zeitschrift* 'revue A'

Le dictionnaire général Duden (Duden 1989) mentionne les composés suivants formés sur *Zeitschrift* comme constituant B: *Fachzeitschrift* 'r. spécialisée',

[2] Les composés allemands sont fondamentalement de structure binaire, quel que soit le nombre des unités lexicales dont ils sont constitués, et toute analyse, à l'excetion de quelques rares structures ternaires 'type bleu-blanc-rouge', peut donc être ramenée à l'étude de la relation entre un constituant A et un constituant B.

Frauenzeitschrift 'r. féminine' et *Literaturzeitschrift* 'r. littéraire'. La liste de noms de l'IDS et la liste de mots inverse de ce même institut (Rückläufige Wortliste, 1986) comprennent en plus les mots suivants: *Gewerkschafts-* 'r. syndicale', *Kinder-* 'r. pour enfants', *Kunden-* 'r. pour les clients', *Mode-* 'r. de mode', *Rundfunk-* 'r. radiophonique', *US-* 'r. américaine', *Verbands-* 'r. de l'association', *Verbraucher-* 'r. de/pour les consommateurs', *Vertriebenen-* 'r. de rapatriés', *Vierteljahres-* 'r. trimestrielle', *Werk-* 'r. d'entreprise, r. «maison»', *Wirtschafts-* 'r. économique', *Wissenschafts-* 'r. scientifique', *Wochen-* 'r. hebdomadaire'. On pourrait sans aucune difficulté allonger indéfiniment la liste en fabriquant des composés sur la base *Zeitschrift* qui auraient l'air tout aussi authentiques que ceux-ci, qui sont attestés dans des textes.

Dans SK on trouve une typologie détaillée des relations à l'intérieur de mots tels que ceux formés sur *Zeitschrift*. Les auteurs caractérisent chaque mot par son type et les rôles sémantiques des constituants A et B. Le type, ou la relation, est explicité au moyen d'une paraphrase. Le composé *Tierbuch* 'livre sur les animaux', qui me semble correspondre du point de vue de sa structure à *Literaturzeitschrift* (revue littéraire) est analysé de la façon suivante (SK 132, 383–95):

Il appartient au groupe des «Bezugskomposita» de type «referentiell», définis par la relation «[B] betrifft/ bezieht sich auf [A]» 'B se rapporte à/ concerne A'. Le premier sous-ensemble, représenté par *Tierbuch*, correspond aux rôles sémantiques «thème / contenu – forme d'expression / représentation». Cette analyse me paraît tout à fait acceptable pour le cas étudié; et on pourrait aussi l'appliquer à l'analyse de *Literaturzeitschrift*: thème/contenu: la littérature; forme de l'expression 'nous dirions plutôt: support': revue. Si l'on analyse aussi en suivant la méthode des auteurs de SK d'autres termes formés sur *Zeitschrift*, comme *Wochenzeitschrift* ou *Werkzeitschrift*, on obtiendrait les résultats suivants:
Wochenzeitschrift 'r. hebdomadaire': date – objet/moyen concerné (SK 499), avec une mention spéciale pour «date», qui devient [itératif] (SK 501).
Werkzeitschrift 'r. d'entreprise': le type le plus proche semble être «agentif/ auteur» (SK 553), défini par les rôles sémantiques agent 'auteur' – produit, bien que l'on puisse aussi analyser «revue/journal d'entreprise» comme «destiné à".

Un mot correspondant à un désigné unique, comme «revue» dans le cas qui nous occupe, va donc se trouver être investi de rôles sémantiques très

variables selon les combinaisons dans lesquelles il va entrer par rapport au composant A. Le mode de fabrication des composés de l'allemand ne donne en général aucune indication sur la relation sémantique ou syntaxique existant entre les composants, contrairement à ce qui se passe en français pour les termes correspondants, où le choix de la préposition ou de la forme du terme modifieur permet de spécifier la relation: revue musicale, pour les jeunes, de mode, sur les animaux etc. Or, ces rôles sémantiques possibles sont inscrits dans les «gènes» du désigné. Une revue fait partie de la classe d'objets «publications», caractérisée par les propriétés suivantes:

.support/forme de l'objet (format, matière, qualité . . .)
.fréquence de la parution (régulière, mensuelle . . .)
.lieu/origine de la publication (française, régionale . . .)
.éditeur/auteur (personne, éditeur, institution, collectivité)
.public visé (jeunes, consommateurs)
.sujet/thème (scientifique, musicale, d'information . . .)

Chacune de ces propriétés peut être actualisée au moyen d'unités lexicales compatibles avec la spécification visée:

.format: *DIN-A4* . . .
.qualité: *Luxus* . . .
.fréquence: *Wochen*, *Vierteljahr*
.lieu: *US*
.éditeur / auteur: *Verband*, *Gewerkschaft*
.public visé: *Verbraucher*, *Kinder*
.sujet / thème / domaine: *Mode*, *Wissenschaft*, *Literatur*

Les propriétés sémantiques du premier constituant permettront de décider quelle est la relation qui est actualisée. Cela ne lève pas toutes les ambiguités. Prenons l'exemple de *Frauenzeitschrift* (mot à mot «femmes-revue»). Le premier constituant *Frauen* peut théoriquement désigner l'éditeur 'revue de femmes, faite par des femmes', le public visé 'revue féminine', le sujet/thème 'revue parlant des femmes'. La troisième interprétation paraît peu probable, bien que des des revues aient pour thèmes «Eltern» 'parents', «Mutter und Kind» 'mère et enfant'. Les ambiguïtés surgissent donc quand un terme peut permettre d'actualiser plusieurs relations, et seule une réflexion pragmatique et une certaine connaissance du monde permettent de donner à une interprétation une plus grande vraisemblance.

2.2. L'exemple de *A-tisch* 'table A'

Les composés formés sur *Tisch* rencontrés dans des dictionnaires ou des textes font appel aux diverses propriétés de l'objet *Tisch*. Ces propriétés découlent en partie de la classe d'objets et du domaine dont il fait partie, à savoir meuble et mobilier (Gross 1992; Mathieu-Colas 1993).

.destination:
à une certaine pièce: *Küchentisch* 't. de cuisine'
à une certaine occupation: *Schreibtisch* 'pour écrire = bureau', *Eßtisch* 'pour manger = de salle à manger . . . ', *Kaffetisch* = 'pour prendre le café, ou du petit déjeûner'
à certains objets: *Blumentisch* 't. pour les fleurs'
à certains occupants: *Vorstandstisch* 't. du directoire', *Stammtisch* 't. des habitués'
.matériau: *Marmortisch* 't. en marbre'
.forme: *Rundtisch* 't. ronde'
.modification: *Klapptisch* 't.pliable, repliable', *Ausziehtisch* 't. à allonges'
.style: *Régence-Tisch* 't. Régence'

L'objet *Tisch* partage certaine de ces propriétés avec d'autres objets du même domaine et de la classe d'objets «meubles», mais il n'en partage pas certaines autres avec la classe d'objets «meubles de rangement» par exemple. Pour décider de l'interprétation à donner à la relation A-*Tisch*, il est nécessaire de pouvoir déterminer quel type de relation ouverte par *Tisch* A peut satisfaire. Selon que A est un terme désignant une activité, des humains, des matériaux, un adjectif de forme ou de matière, c'est l'une ou l'autre des relations mentionnées qui sera actualisée.

2.3. L'exemple de A-*methode* 'méthode A'

La liste des noms de l'IDS contient 72 mots complexes se terminant par *-methode/methoden*. Le désigné *Methode* inclut les propriétés suivantes:

.application à une certaine activité/opération/tâche:
Färbemethode 'm. de teinture', *Werbemethode* 'm. publicitaire': méthode pour, méthode de, méthode adj.
.utilisation d'une certaine procédure: *Quotenmethode* 'm. des quotas', *Buchstabiermethode* 'm. qui consiste à épeler pour apprendre à lire', *Temperaturmethode* 'm. des températures': méthode de, des, qui consiste en.

.est utilisée par certaines personnes, en certains lieux: *Gangstermethode* 'm. de gangsters', *Wildwestmethode* 'm. du Far-West': méthode de, utilisée par.
.a été développée par X: *Knaus-Ogino-Methode* 'méthode X'
.autre caractéristiques: *Alternativ-Methode* 'm. alternative', *Schnell-Methoden* 'méthodes rapides': m.Adj.

On constate que certains A peuvent être interprétés aussi bien comme activité que comme procédure: *Aufpulverungsmethode* 'm. de pulvérisation' est susceptible théoriquement de signifier: méthode pour pulvériser, destinée à, que méthode qui consiste à pulvériser.

2.4. L'exemple de A-*technik* 'technique A'

La liste de l'IDS comprend plusieurs centaines de termes formés sur *-technik*. Le désigné de *Technik* est proche de celui de *Methode* et l'on trouvera donc des combinaisons A-*technik* qui correspondent à la même relation que A-*Methode*:

.pour une certaine activité: *Bautechnik* 't. de construction', *Behandlungstechnik* 't. de traitement': technique de.
.utilisation d'une certaine procédure: *Digitaltechnik* 't. digitale': adj.
.domaine concerné: *Atomtechnik* 'adj: atomique', *Bühnentechnik* 'adj: scénique, de scène'.
.autres caractéristiques: *Schlüsseltechnik* 'technique-clé', *Zukunftstechnik* 'd'avenir'.

Les quelques essais de traduction de la relation 'de, pour, en vue de, qui concerne, destinée à, adj, qui relève de . . . ' montrent que chaque relation correspond à un certain type de paraphrase en français, paraphrase nécessaire puisque le français ne connait pas au même degré que l'allemand la formation de composés par pure juxtaposition: *revue femmes, *technique atome, *méthode gangsters, bien que cette forme soit semble-t-il en train de se développer. Il paraît donc utile, dans la perspective du traitement et de la traduction automatiques, d'analyser les propriétés combinatoires des diverses unités lexicales de la langue, d'en dresser l'inventaire pour chaque unité en fonction de la classe d'objets et du domaine à laquelle elle appartient, et d'exprimer ces propriété combinatoires au moyen de paraphrases appropriées. Les auteurs de SK ont bien signalé l'existence de classes reposant sur les caractéristiques sémantiques de la base B, et notamment le grand nombre de composés formés sur des abstraits au contenu dénotatif réduit (SK 31–33), mais ils n'en tirent pas toutes

les conséquences, à savoir la nécessité de dresser le tableau de la combinatoire de chaque terme.[3]

3. L'assymétrie de la relation A–B

Dans tous les types de mots complexes qui viennent d'être analysés, il y a une relation d'inclusion entre A–B et B, c'est-à-dire que dans tous les cas «AB est un A»: *ein Marmortisch ist ein Tisch.* C'est le terme B qui définit les combinaisons AB possibles et non l'inverse. La relation AB est induite par les caractéristiques de B et par ce que les locuteurs savent de B. Une unité lexicale en position B n'entre donc pas dans les mêmes combinaisons qu'en position A, puisque dans ce dernier cas, c'est l'autre unité qui définit la combinatoire possible. Je vais illustrer cette assymétrie des relations A–B selon que l'unité lexicale est placée en position A ou en posiiton B à l'aide du mot *Daten* 'données'. Les exemples sont également tirés du corpus de l'IDS:

3.1. *Daten*-B

.*Datenaustausch* 'échange de d.', -*eingabe* 'entrée de d.', -*erfassung* 'enregistrement de d.', -*erhebung* 'collecte, relevé de d.', -*mißbrauch* 'abus de d.' etc . . . B est un nom prédicatif souvent déverbatif qui désigne une opération: collecter, entrer, relever, saisir etc . . .
.*Datenschützer* 'personne chargée de veiller à la protection des d.', -*typistin* 'personne qui entre les d.' etc . . . B est un nom d'agent dérivé de verbe ou d'abstrait, désignant un opérateur humain, qui exerce une certaine activité, exprimée par la base verbale ou abstraite: protéger, taper, entrer.
.*Datenspeicher*: B est un nom d'outil déverbatif, désigne l'outil et sa destination.
.*Datentechnik* 'technique concernant les d.': B est un nom abstrait, désignant une certaine procédure (cf. **2.4**).
.*Datenbank* 'banque de d.', -*basis* 'base de d.', -*netze* 'réseaux de d.', -*bänder* 'bandes magnétiques de d.', -*träger* 'supports de d.', -*zentrale* 'centrale de d.': B désigne un support, un mode de structure ou d'organisation de A.
.*Datenbestände* 'ensemble de d.' -*fluß* 'flux de d.', -*menge* 'ensemble de d.': B est un terme général quantifieur. Ce n'est plus A qui détermine B, mais

[3] Je laisserai de côté dans cette communication le cas des composés formés sur un composant B de type «relationnel / rectionnel» ou prédicatif. Dans ce cas, ce sont les propriétés combinatoires du terme B qui ouvrent un certain nombe de structures ou de relations possibles, et les caractéristiques sémantique de A permettent de décider de la relation actualisée par une association A–B donnée (Bresson 1991).

l'inverse, ainsi qu'on peut le constater en remplçant l'ensemble du mot par A: *Datenmaterial*, *Datenmenge*: *Daten*.

Les composés dans lesquels *Daten* est en position A contiennent en position B (à l'exception des quantifieurs' des mots désignant une procédure, une technique, un agent, un outil, un mode de support ou d'organisation. Ils correspondent aux opérations possibles sur l'objet désigné par *Daten* ou sur un autre objet partageant les mêmes propriétés.

3.2. A-*daten*

Umfrage- 'd. de sondage', *Volkszählung-* 'chiffres du recensement', *Wetter-* 'd. météorologiques' etc . . . : dans toutes ces occurrences, A définit une sous-classe de *Daten* et indique la nature ou l'origine de ces données. Peut se combiner avec *Daten* quelque chose susceptible d'être mesuré et de donner un ensemble de données chiffrées.

On voit ainsi se dessiner deux types de paradigmes, dont le principe d'organisation est rigoureusement différent selon que le terme de base du paradigme (ici: *Daten*) se trouve en position A ou en position B. Le paradigme *Daten*-B correspond à tout ce qui peut être dit ou fait concernant *Daten*, le paradigme A-*Daten* correspond à la spécification des divers types de *Daten*. La fonction de A et de B dans la séquence A–B n'est pas de même nature. B ouvre un certain nombre de combinaisons possibles, A vient se loger dans l'une des combinaisons ouvertes par B. La structure linéaire des composés allemands est particulièrement propice à l'interprétation qui part de A pour aller vers B, c'est-à-dire tout ce qui peut être dit de A, alors que le français fonctionne en sens inverse et privilégie B comme point de départ de la relation.

4. Classes sémantico-syntaxiques[4]

La nature de la relation actualisée entre A et B dépend donc en premier lieu des propriétés combinatoires de ces constituants, qui sont elles-mêmes déterminées par l'appartenance de l'élément à une classe sémantico-syntaxique. C'est à ce classement sémantique que se sont attelés pour l'allemand des auteurs comme Dornseiff (Dornseiff 1933, 1970', qui propose de classer conceptuellement les mots de l'allemand selon des classes d'objets («begrifflich nach Sachgruppen ordnen»). Mais ce classement, s'il peut certainement être en partie repris, ne peut pas être utilisé tel quel, car il n'a qu'une base conceptuelle et repose

[4] Pour une représentation d'ensemble de cette question cf. Bresson (1992).

essentiellement sur le champ sémantique auquel appartient un mot. Il correspond plus à la définition de domaines (cf. Mathieu-Colas 1993) que de classes d'objets caractérisées par des propriétés linguistiques précises. On trouve dans le chapitre 13 (Dornseiff 363–82) consacré aux signes, à la communication et à la langue, un sous-chapitre 13.6 réservé à «informer, porter à la connaissance de» 'bekanntmachen', dans lequel on trouve les verbes, les noms, les adverbes, les expressions figées et imagées relevant de ce champ sans indication de la classe syntaxique. L'indication du domaine conceptuel est précieuse, mais elle n'est pas suffisante, il faut aussi savoir si le mot désigne un objet, un prédicat, un agent, un quantifieur etc.

Il faut donc aller plus loin dans dans la recherche et la description des classes. Une direction de recherche est fournie par G. Gross (Gross 1990, 1992), qui établit des listes les plus exhaustives possibles de mots d'une même classe lexico-syntaxique 'noms de fleurs, de boissons, d'actes juridiques etc.). Les mots d'une même classe ne partagent pas seulement des propriétés sémantiques, mais aussi des propriétés syntaxiques.

Mais il est nécessaire, parallèlement à cette recherche sur les classes lexicales terminales, de définir une typologie syntactico-sémantique: les notions de «substantif abstrait ou de substantif prédicatif» ont, en dehors de leur valeur sémantique, une signification pour la syntaxe de ces éléments. Un substantif prédicatif a une structure d'arguments donnée: Une blessure est «infligée par quelqu'un à quelqu'un, en telle ou telle occasion, sur telle ou telle partie du corps, au moyen de telle ou telle arme» etc . . . , et les composés nominaux allemands formés sur *Verletzung* 'blessure' seront interprétés comme blessure à (*Kopf-* 'tête'), blessure de (*Krieg-*, 'guerre'), blessure par (*Kugel-*, 'balle') selon que le terme A appartient à la classe des parties du corps, des événements ou activités, des armes ou objets (Bresson 1991, 186) et est susceptible, en fonction de cette appartenance, d'occuper telle ou telle position d'argument par rapport au prédicat *Verletzung*.

5. Exemple de classe sémantico-syntaxique: la quantification

Un certain nombre de composés nominaux allemands non lexicalisés sont caractérisés par le fait que l'élément B, au lieu d'être déterminé par l'élément A, est un nom de nombre, de quantité ou de mesure, qui opère comme quantificateur sur A; il peut s'agir de noms désignant des unités d'une certaine matière, tels que *Korn* 'grain' ou *Molekül* 'molécule', de noms désignant une certaine quantité de X ou un ensemble de X, tels que *Menge* 'foule, ensemble' ou *Gruppe* 'groupe'. On peut identifier un certains nombre d'opérations quan-

tifiantes, susceptibles d'être exprimées par un ensemble d'unités lexicales qu'il importe de répertorier et d'affecter dans le dictionnaire d'un symbole servant à signaler leur valeur d'opérateur quantifieur; il existe souvent un ou plusieurs mots classifieurs représentatifs de tout le groupe. Les opérations de quantification sur un objet X sont variées. Il peut s'agir de désigner une unité ou occurrence de X, un sous-ensemble de X, une certaine quantité comptable ou non de X, de le mesurer, de le comparer, d'en mesurer les variations etc. Dans un premier temps, je propose la liste des opérateurs quantifieurs suivants:

.QUN (quantifieur unité)= une unité de A; mots classifieurs: *Einheit*, *Element*; ex: *Produktionseinheit* 'unité de production', *Bauelement* 'unité de construction'; autres mots appartenant à cette classe: *Atom*, *Ion*, *Keim* 'germe', *Molekül*, *Organismus*, *Partikel*, *Teilchen* 'particule', *Zelle* 'cellule' etc.
.QSTR (quantifieur structurant)= une structure de A; ces mots servent à désigner la structure habituelle d'apparition d'une matière donnée; mots appartenant à cette classe: *Schicht* 'couche', *Vorkommen* 'gisement', *Staude* 'pied de', *Stück* 'morceau de, pour le sucre par exemple' etc.
.QOCC (quantifieur occurrence de)= un cas de A; mot classifieur: *Fall* 'cas de'; ex: *Ausnahmefall* 'exception', *Krankheitsfälle* 'des cas de maladie'; autres termes: *Phänomen*;
.QPAR (quantifieur partie de)= une partie non structurée de A; mot classifieur: *Teil* 'partie de'; ex: *Bevölkerungsteil* 'partie de la population'; autres termes: *Spur* 'trace', *Anteil* 'part', *Abschnitt* 'passage', *Bruchteil* 'parcelle', *Splitter* 'bris', *Stelle* 'passage'.
.QSSTR (quantifieur sous-structure)= une partie structurée de A; mots classifieurs: *Glied* 'membre', *Teil* 'pièce, partie de'; ex: *Motorteil* 'pièce du moteur' autres termes: *Absatz* 'paragraphe', *Kern* 'noyau' etc.
.QGR (quantifieur groupe de)= un certain nombre d'unités A dénombrables; mot classifieur: *Gruppe*; ex: *Kindergruppe* 'groupe d'enfants'; autres termes 'il y en a plusieurs centaines': *Schar* 'grand nombre', *Menge* 'foule' etc.
.QQ (quantifieur une certaine quantité de)= une quantité non mesurée de A non comptable; mot classifieur: *Masse*; ex: *Lavamassen* 'masses de lave'; autres termes: *Menge*, *Haufen* 'tas'.
.QN (quantifieur nombre de): spécifie une valeur numérique de A quantifiable; mot classifieur: *Zahl*; ex: *Übersiedlerzahl* 'nombre de rapatriés'; autres termes: *Werte* 'valeurs', *Ziffern* 'chiffres', *Dosis* 'dose', *Statistik* etc.
.QCOL (quantifieur totalité ou collectif): désigne la totalité de A; mots classifieurs: *Gesamtheit* 'totalité', *Bestand* 'l'ensemble'; ex: *Fichtenbestand*

'l'ensemble des épicéas'; autres termes: *Allgemeinheit*, *Ganze* 'ensemble', *Ladung* 'chargement', *Vorrat* 'réserve', *Komplex* 'ensemble' etc.

.QM (quantifieur mesure): indique les mesures de A; il se subdivise en plusieurs opérations selon le type d'indication de mesure correspondant à l'objet désigné par A; chaque dimension/mesure est exprimée au moyen d'unités de mesure appropriées.

.QDIM (quantifieur dimensions): désigne les dimensions de A; mots classifieurs: *Ausmaße*, *Dimension* 'dimensions', *Umfang* 'volume'; ex: *Zimmerdimensionen* 'dimensions de la pièce'; autres termes: *Höhe* 'hauteur', *Breite* 'largeur', *Länge* 'longueur', *Fläche* 'surface' etc.

.QH (quantifieur niveau): désigne le niveau, le taux de A; mots classifieurs: *Niveau*, *Pegel* 'niveau'; ex: *Ozonpegel* 'niveau d'ozone'; souvent exprimé en valeur relative et non absolue comme QDIM; autres termes: *Satz* 'taux', *Rate* 'pourcentage', *Höhe* 'hauteur' etc.

.QSOM (quantifieur somme): désigne une certaine somme de A, souvent monétaire; mot classifieur: *Summe* 'somme'; ex: *Investitionssumme*; autres mots classifieurs: *Betrag* 'montant', *Zuschuß* 'subvention', *Gebühr* 'taxe', *Abgabe* 'prélèvement' etc.

.QD (quantifieur temporel durée): s'applique à des A quantifiables par leur durée; mot classifieur: *Dauer* 'durée'; ex: *Geltungsdauer* 'durée de validité'; autres termes: *Zeit* 'temps, durée', *Periode* etc.

.QFR (quantifieur temporel fréquence, vitesse): indique la fréquence ou le rhythme ou la vitesse de A; mot classifieur: *Frequenz*, *Rhythmus*, *Geschwindigkeit* 'vitesse'; ex: *Arbeitsrhythmus* 'rythme de travail'; autres mots: *Tempo*, *Häufigkeit* etc.

.QDIS (quantifieur spatial distance): concerne des A de valeur spatiale; mot classifieur: *Entfernung*; ex: *Grenzentfernung* 'distance de la frontière'; autres termes: *Distanz*, *Abstand* 'espace, distance' etc.

.QS (quantifieur scalaire, intensité): indication du degré, de l'intensité de A; mots classifieurs: *Grad* 'degré', *Intensität*; ex: *Verstrahlungsgrad* 'degré d'irradiation'; autres termes: *Skala* 'échelle', *Stufe* 'degré' etc.

.QDIF (quantifieur différence, comparaison): mesure la différence entre des valeurs de A; mot classifieur: *Unterschied* 'différence'; ex: *Höhenunterschiede* 'différences d'altitude'; autres termes: *Gleichheit* 'égalité', *Ungleichheit* 'inégalité' etc.

.QV (quantifieur variations): spécifie les variations de A; mot classifieur: *Variation*, *Schwankung*; ex: *Klimavariationen*; autres termes: *Verlust* 'perte', *Abnahme* 'diminution', *Erhöhung* 'augmentation' etc.

6. Conclusion et perspectives

L'exemple de la quantification, qui représente une relation assez bien identifiable entre deux composants A et B dans le cas de composés nominaux de l'allemand, avait pour objectif d'illustrer comment l'établissement de classes sémantico-syntaxiques dans les unités lexicales répertoriées d'une langue peut être utilisé pour déterminer la relation sémantique susceptible d'être réalisée lors de la mise en connexion de deux unités de la langue. Les composés non lexicalisés de l'allemand sont un cas extrême, puisque dans ces constructions aucune marque morphologique ne permet de déduire le type de relation syntaxique existant entre A et B. Il s'agit d'une pure concaténation, et la mise en rapport des deux unités par la concaténation est le seul indice apparent utilisable pour déchiffrer la valeur de la construction. Mais cet indice a une valeur si générale qu'il est inutilisable. Il signifie simplement: «il y a une relation entre A et B». D'où la nécessité de chercher, dans les propriétés de A et de B, les éléments de signification qui peuvent justifier que ces deux unités lexicales soient mis en relation.

Cela suppose un très important travail de description des unités lexicales d'une langue, de l'allemand dans le cas qui nous occupe, qui permette de prévoir leurs propriétés combinatoires. Ce travail doit comporter un aspect sémantico-syntaxique et un aspect encyclopédique. L'aspect sémantico-syntaxique constitue le point de départ de l'analyse. Un nom prédicatif, déverbatif ou non, est susceptible d'être associé avec des arguments qui sont mis en évidence lors de l'emploi de ce mot en position prédicative ou sous forme verbale: *eine Kopfverletzung: jmd hat eine Verletzung am Kopf*. Ces indications doivent figurer dans des dictionnaires qui donneront toutes les indications nécessaires sur les propriétés syntaxiques des unités lexicales. Mais il est aussi indispensable d'indiquer, à côté de la définition sémantique d'un mot son appartenance à un domaine et à une classe d'objets. C'est ce qui permettra de décider s'il peut ou non se combiner avec une autre unité lexicale. Le travail est énorme, mais c'est à ce prix que l'on peut espérer certains progrès dans le traitement automatique de langues semi-agglutinantes comme l'allemand.

Bibliographie

Bresson, D. 1991. Zur Analyse nominaler relationaler Komposita im Deutschen im Hinblick auf die maschinelle Sprachverarbeitung. In: Cahiers d'Etudes Germaniques 21: 179–88. Aix-en-Provence.

Bresson, D. 1992. La relation syntaxique et sémantique dans les composés nominaux de l'allemand. In: Systèmes interactifs. Mélanges en l'honneur de Jean David, 67–79. Metz.

Brückner, T. 1986. Refer, Benutzerhandbuch. Institut für deutsche Sprache, Mannheim.

Duden 1978. Das große Wörterbuch der deutschen Sprache in 6 Bänden. Bibliographisches Institut, Mannheim.

Duden 1989. Deutsches Universalwörterbuch A–Z. Bibliographisches Institut, Mannheim.

Gross, G. 1988. Degré de figement des noms composés. In: Langages 90: 57–72.

Gross, G. 1990. Les classes d'objets. In: Actes du colloque sur les industries de la langue. Montréal.

Gross, G. 1992. Forme d'un dictionnaire électronique. In: Clas, A.–Safar, H. (éd.): L'environnement traductionnel, 255–71. Presses de l'Université du Québec, Sillery (Canada).

Kürschner, W. 1974. Zur syntaktischen Beschreibung deutscher Nominalkomposita. Tübingen.

Mathieu-Colas, M. 1993. Dictionnaire électronique des mots français à trait d'union. Thèse de doctorat en linguistique, Université de Paris XIII.

Rückläufige Wortliste zum heutigen Deutsch, 2. Auflage 1986. Bearbeitet von T. Brückner und C. Sauter. Institut für deutsche Sprache, Mannheim.

Seidel, E. 1988. Wörterbuch Umweltschutztechnik. Verlag Harri Deutsch, Thun und Frankfurt/Main.

SK 1991. Deutsche Wortbildung, Vierter Hauptteil: Substantivkomposita. De Gruyter, Berlin–New York.

L'adresse de l'auteur: Daniel Bresson
Laboratoire de Linguistique Germanique
Université d'Aix-Marseille I
F–13621 Aix-en-Provence Cedex
France

Acta Linguistica Hungarica, Vol. 41 (1–4), pp. 109–121 (1992–93)

USING A BILINGUAL COMPUTERIZED DICTIONARY TO RETRIEVE SUPPORT VERBS AND COMBINATORIAL INFORMATION

THIERRY FONTENELLE

Introduction

Lexical acquisition has become a crucial research topic in computational linguistics and many researchers are convinced that it is not desirable to start coding thousands of lexical entries from scratch. Therefore the idea of re-using already-existing lexical resources (dictionaries or large textual corpora) has emerged. But using machine-readable dictionaries (MRDs) to feed the lexical component of NLP systems requires careful study of the microstructure of the dictionary and of the coding devices adopted by the lexicographers (cf. Boguraev 1991; Calzolari 1989; Fontenelle 1992a). In this paper, I wish to show that the machine-readable version of a bilingual dictionary, namely the **Robert & Collins English–French French–English dictionary** can be used to extract co-occurrence knowledge and information about support verbs. The results of the experiment described here are then compared to other methods advocating the use of statistical tools and large textual corpora to extract the same type of information (Smadja 1991; Church–Hanks 1990).

Co-occurrence relations

In a paper originally presented at the First Lexical Acquisition Workshop in Detroit, Smadja (1991) describes XTRACT, a co-occurrence compiler that retrieves lexical relations from a large statistically analysed corpus. The lexical relations XTRACT acquires are co-occurrence relations, a.k.a. idiosyncratic collocations. This type of information is extremely important for language generation since it makes it possible to encode lexical constraints that account for the well- or ill-formedness of the following sentences:

Akadémiai Kiadó, Budapest

(1) John quenched the fire.
(2) John extinguished the fire.
(3) John quenched his thirst.
(4) *John extinguished his thirst.
(5) John slaked his thirst.
(6) *John slaked the fire.

The term "collocation" refers to the syntagmatic combination of lexical items. These constraints need to be encoded in order to avoid oddities in the generation process (cf. Smadja's examples: **a powerful tea* instead of *a strong tea*; **a strong car* instead of *a powerful car*). Since collocational restrictions are unpredictable, they need to be encoded either manually (by a team of lexicographers), which is both time-consuming and costly, or (semi-)automatically.

Smadja's XTRACT compiler extracts co-occurrence knowledge from very large textual corpora by identifying statistically relevant lexical relations. To quote Smadja's paper, "XTRACT takes as input a corpus d and a dictionary specifying parts of speech. It produces a list of tuples (w1, w2, cook-info), where (w1, w2) is a lexical relation between two open-class words (w1 and w2) identified in d, and cook-info is a set of statistical figures representing the lexical relations within the distribution of words collocating with w1" (e.g. decision_make_21.7657; decision_take_5.321). The following examples illustrate some results for two corpora consisting of approximately 2,300,000 words. Each row represents a productive word and its collocates are classified according to the part of speech (N=noun; V=verb; A=adjective).

law:	V: change, enforce, pass, violate
	N: court, universe
	A: Jewish, penal, physical
argument:	V: have, reject, settle, use
	N: court
	A: side, valid
food:	V: eat, have, prepare, sell
	N: health, industry, product, shortage, supply
	A: good, kosher, spicy, Yemenite

Smadja has then refined his algorithm to be able to specify that a noun used as an object of a verb is a direct or an indirect object (taking into account the distribution of words, the presence of relative clauses or passive constructions, etc.). As can be seen above, this program seems to be mainly efficient

with respect to adjective–noun, noun–noun and verb–noun lexical relations. In what follows, I would like to demonstrate that this type of information can be enriched with co-occurrence knowledge extracted from the Robert & Collins dictionary.

Collocations in the Robert & Collins dictionary

The magnetic tapes of the Robert & Collins English–French French–English dictionary (Atkins–Duval 1978) were made available to our department for research purposes under contract with the publishers. The WordCruncher Text Retrieval Software package (running under MS-DOS) was chosen to exploit the content of the dictionary. WordCruncher has generated a general index (with the frequency of occurrence) of all the words that appear in the dictionary. The English words now appear in capital letters while the French words appear in small letters. The metalinguistic information that appears in italics in the printed version now appears between angled brackets (parts of speech, subject fields, co-occurrence information . . .).

A systematic approach has been adopted by the lexicographers to account for collocational constraints (in italics in the printed version, between angled brackets in our processed file):

- typical noun subjects of the verb headword appear in square brackets [ ];
- typical noun complements of the noun headword appear in square brackets [ ];
- typical noun objects of transitive verbs are unbracketed;
- typical nouns modified by an adjective are unbracketed.

The following examples (from the English–French part) illustrate this approach:

abolish *vt practice, custom* supprimer; *death penalty* abolir; *law* abroger, abolir
do away with *vt fus* (a) *custom, law, document* supprimer; *building* démolir
forceful *adj person, character* énergique; *argument, reasoning* vigoureux, puissant
go through 1 *vi [law, bill]* passer, être voté; *[business deal]* être conclu, être fait, se faire
incontrovertible *adj fact* indéniable; *argument, explanation* irréfutable; *sign, proof* irrécusable
infringe 1 *vt obligation* contrevenir à; *law, rule* enfreindre, transgresser, contrevenir à

penal *adj law, clause* pénal; *offence* punissable
revocation *n [order, promise, edict]* révocation; *[law, bill]* abrogation; *[licence]* retrait; *[decision]* annulation
validate *vt claim, document* valider; *argument* prouver la justesse de

One of the major drawbacks of printed dictionaries is that they only provide users with one access path, namely the alphabetical order. The Word-Cruncher organisation of our MRD, on the contrary, enables us to instantaneously retrieve all the occurrences of a given word in italics, together with the headword under which it is found. A program can then assign a syntactic/semantic link to the pair of collocates retrieved from the dictionary. This link is assigned automatically on the basis of typographical information combined with the part of speech of the headword. If we apply this program to the examples above, focussing on the collocates of *law* and *argument*, the output is:

law:	adjective: penal
	subject_of: go through
	object_of: abolish, do away with, infringe
	modifier_of_noun: revocation
argument:	adjective: forceful, incontrovertible
	object_of: validate

If we start from the hypothesis that co-occurrence knowledge is spread across the entire MRD, we can retrieve words with their collocates and tag the pair of items with a label that accounts for the surface link between the members of the pairs. If we consider that 89 items co-occurring with *law* can be retrieved from the dictionary (+ 115 items collocating with *argument*; 216 items for *food*, etc.), we can reasonably conclude that the information contained in the dictionary might complement the data obtained from corpus-based analyses (the transitive verbs that can take *law* as direct object are, according to Robert & Collins: *abolish*, *annul*, *carry out*, *circumvent*, *contravene*, *defy*, *disobey*, *do away with*, *elude*, *enact*, *enforce*, *establish*, *evade*, *get round*, *infringe*, *invoke*, *keep*, *neglect*, *obey*, *offend against*, *put into operation*, *override*, *promulgate*, *reform*, *repeal*, *rescind*, *respect*, *revoke*, *sanction*, *stretch*, *subvert*, *trespass against*, *uphold* and *vote in*).

Although this paper focuses on the acquisition of English collocations, it goes without saying that the method which I suggest here can also be applied to the French–English part of the dictionary. Since French lexicography has not given birth to *commercially-available* dictionaries that would provide as

rich a source of lexical information as some English learner's dictionaries do, the possibility of extracting such knowledge for French from the Robert & Collins dictionary should certainly not be dismissed. It would for example reveal that the word "loi" (=law) can collocate with transitive verbs such as *abolir*, *adopter*, *appliquer*, *approuver*, *concocter*, *contrevenir à*, *édicter*, *éluder*, *invoquer*, *obéir à*, *faire opposition à*, *réformer*, *respecter*, *ressusciter*, *sanctionner*, *supprimer*, *toucher à*, *violer* or *voter*.

Polysemy in collocations

Analysis of the list of words occurring as typical objects/subjects/head nouns in italics reveals that they are most often used in their basic, prototypical meaning. There may be cases, however, where a given polysemous word is used by the lexicographer in its various senses. This accounts for the distinction we have to make in analysing the list of verbs associated with the French noun *prix* in the French–English part of the dictionary. *Prix* is indeed ambiguous and can refer to either Eng. *price* or *prize* and the verbs that collocate with this noun usually depend on its meaning. The list of verbs should therefore be split into two sublists (Fontenelle (1992b) describes in greater detail the collocations of the English noun *price* extracted from the dictionary and from a large corpus):

$prix_1$ (=prize): attribuer, avoir, décerner, décrocher, donner, emporter, remporter . . .
$prix_2$ (=price): augmenter, baisser, débloquer, dégringoler, diminuer, s'effondrer, geler, grimper, majorer, plafonner, réduire . . . (sample lists)

This means that caution should be exercised in extracting collocations since word meaning obviously plays an important part in the use that is made of collocations. If co-occurrence knowledge can be acquired automatically from the dictionary, the semantic interpretation requires human intervention and the first step is the disambiguation of the base of the collocation. It can also be noted that this interpretation by a lexicographer is also required when collocations are extracted from corpora (cf. Smadja's comment in section 3 of his paper: "Pure lexical relations are a statistical phenomenon and can thus be acquired automatically; similarly the syntactic interpretation can be automated. In contrast, the *semantic interpretation* requires human intervention and is domain dependent"—italics mine). The following sections sketch various approaches that can be adopted to do this semantic interpretation.

Lexical functions

Examining the list of lexical collocations for *law* reveals that the relationship between the base and the collocator is variable. *Enact*, *establish*, *promulgate* and *vote in* can be considered as near synonyms whereas *abolish*, *annul*, *repeal*, *rescind* and *revoke* all express the opposite meaning. The first set refers to what Benson *et al.* (1986) call CA collocations (i.e. verbs denoting creation and/or activation). The second set of items refers to EN collocations (verbs meaning eradication and/or nullification).

Such differences can also be accounted for in the framework of Mel'čuk's Meaning-Text Theory (cf. Mel'čuk–Žholkovsky 1988; Steele 1990). The most important component of this theory is the Explanatory Combinatory Dictionary (ECD). This dictionary is called combinatory because it is intended to display the combinatorial properties of words (Apresyan *et al.* 1969). To do so, it resorts to a well-defined set of **lexical functions** that express a meaning relationship between a keyword and other words with which it frequently co-occurs. The typical example given in the MTT/ECD literature is the **Magn** lexical function, which expresses the relationship between a phenomenon and the highest degree of this phenomenon. For example, *Magn (pain) = excruciating* means that the adjective *excruciating* has to be used to express a very intense pain (other items such as *gnawing*, *keen*, *searing*, *sharp* ... can also be used to convey this meaning). Mel'čuk has identified approximately 60 lexical functions, ranging from traditional lexical-semantic relations (Syn=synonym; Anti=antonym; Gener=hypernym) to lesser-known relations (Son = typical sound of - as in Son (dog) = bark; Liqu = liquidate/eliminate—as in Liqu (file) = delete, erase ...).

The lexical collocations extracted from the Robert & Collins dictionary can also be analysed in terms of lexical functions. The CA collocations detailed above can be represented as follows in the ECD framework:

CausFunc_0 (law) = enact, establish, promulgate, vote in

where Caus is the causative operator expressing the creation of something and Func_0 refers to the semantically empty verb which takes the keyword as its subject (to enact a law = to cause a law to come into force).

The EN collocations would be represented in the following way:

LiquFunc_0 (law) = abolish, annul, repeal, rescind, revoke ...

where Liqu expresses the nullification of something.

The Real function is used by Mel'čuk to express the fact that the deep-syntactic actants comply with the requirement of the argument of the lexical function:

Real (law) = carry out, obey, respect

The main problem is that the linguist analysing the pairs of collocates has to identify the lexical function which relates the two items. In the ECD suggested by Mel'čuk, the lexicographer starts with a base (the keyword) and a list of lexical functions. The main task is to discover how a given LF is realized.

Support verbs

The type of data that can be extracted from the Robert & Collins makes it possible to enrich our description of the English lexicon with information about support verbs. Machonis (1991) defines support verbs as verbs that carry little semantic content and are used for syntactic support (cf. *commit suicide*, *make an analysis*). Gross (1981) analyses support verbs within the lexicon-grammar framework and notes that they embody semantic restrictions that are much more complex than selection restrictions. Many deverbal nouns usually need some kind of semantically empty verb which only conveys information about person, tense and aspect, as in:

(7) John brings forward an argument.

John can be seen as the "subject" of *argument*. Support verbs can then be defined in terms of their property to preserve the relationship between the subject and the supported noun. This explains why *accept* cannot be considered as a support verb in:

(8) John accepted the argument.

Accept is not a semantically empty verb. It carries a special meaning together with information about tense, person and aspect. If we substitute *speech* for *argument* in (7), we end up with an unacceptable sentence:

(7') *John brought forward a speech.

The reason is that the verb that can be used to support *speech* is *deliver*:

(7") John delivered a speech.

Other verbs, such as *make*, *give*, *address* or *get out* can also be used as support verbs with respect to *speech*.

As can be seen above, support verbs actually represent a subset of lexical collocations. They will usually be found in the "object_of" class of verbs generated by our program (in the above-mentioned example, four support verbs in our dictionary contain information about possible collocability with *argument*. John *brings forward*, *enforces*, *poses* or *puts forward* an argument).

It should be noted that support verbs roughly correspond to the type of lexical relation that can be encoded through the **Oper** lexical function used by Mel'čuk. $Oper_1$, $Oper_2$ ... refer to the semantically empty verb which takes the first, second ... actant of the keyword as its subject and the keyword as its direct object (Steele 1990, 57). The examples given in Steele's book are:

$Oper_1$ (attention) = pay
$Oper_2$ (attention) = attract.

Analysis of the "object_of" class of *attention* in the Robert & Collins shows that it can be combined with many more verbs:

$Oper_1$ (attention) = concentrate, focus, turn
$Oper_2$ (attention) = arrest, capture, draw, engage, engross, excite, fix, invite, occupy, stir up, take up, win

It could be objected that the link between *pay* and *attention* differs significantly from the link between *concentrate/focus/turn* and *attention*. Indeed, the former combination should be considered more as an idiom (or fixed multiword unit) than as a proper collocation. Although the distinction between idioms and collocations is far from being a clear-cut one (many linguists even argue that it is more a cline than a dichotomy), it should be realized that the expression *pay attention (to sth)* displays some characteristics that are typical of idioms. For example, it cannot undergo several syntactic manipulations that are normally possible with regular collocations:

a) no possibility of using a pro-form or pronominal anaphoric reference:

(9) *She didn't pay attention to me but I paid *some/it* to her.

b) no possible cleft sentence:

(10) *It is attention that should be paid to ...

Moreover, the mere fact that *attention* cannot be modified by a determiner (*He paid *his* attention to...) clearly demonstrates that *pay_attention* is different from *focus_attention* or *concentrate_attention*. The possibility of passivizing the expression (Attention should be paid to...) or inserting some modifying material (They paid *scant/little/considerable* attention to...) shows, however, that it is also different from truly fixed, totally frozen expressions such as *to take place*. Cruse (1986, 41) uses the term 'bound collocations' for combinations such as *pay_attention* which belong to the transitional area bordering on idiom.

Support verbs and machine translation

In a recent paper, Danlos-Samvelian (1992) suggest a methodology to use support verb information in machine translation. Their contention is that it is preferable to start from the predicative noun, which is the most informative element in a sentence, insofar as it is responsible for the selection of the accompanying verbs. The ultimate aim of such an approach is to avoid bilingual context-sensitive rules where the translation of a verb is determined by its object. Danlos–Samvelian (1992, 21) illustrate the traditional approach with French–English examples of such rules:

avoir (_ habitude) ⟶ be in
perdre (_ habitude) ⟶ get out of
prendre (_ habitude) ⟶ get into.

They suggest that these rules are unnecessary and that information at transfer level should be limited to a simple translation rule such as *habitude* ⟶ *habit*. In the monolingual lexicon, however, each noun should be described in terms of the set of support verbs with which it can be associated. Each verb should also be assigned a given semantic feature reflecting the aspectual,

diathetic or modal values conveyed by the combination of the two. Some of the semantic values suggested by Danlos–Samvelian (1992) are:

neuter: be in (_ habit)
inchoative (beginning of a process or state): get into (_ habit)
terminative (end of a process or state): get out of (_ habit)
durative (duration of a process or state): keep (_ ascendancy)
causative (diathetic value—causative denotation): give (_ hangover).

Instead of having numerous (and costly) context-sensitive rules at transfer level, the system would contain a richer monolingual description of lexical items. A noun such as *habit* would be described in the English lexicon as follows:

habit ⟶ {*be in* (neuter), *get out of* (terminative), *get into* (inchoative)}.

A similar entry would characterize *habitude* in the French monolingual lexicon:

habitude ⟶ {*avoir* (neuter), *perdre* (terminative), *prendre* (inchoative)}.

Danlos–Samvelian (1992) then explain how these types of information can be used in a transfer-based MT system such as EUROTRA. They also argue in favour of the automatic extraction of such collocational information from large corpora. The technique I have described above shows, however, that the Robert & Collins dictionary is, at least to some extent, a suitable candidate for extracting such information. Since it contains bilingual data, it is fairly easy to derive the relevant support verbs for English and French. The task of the lexicographer eventually boils down to assigning the correct semantic value to each support verb. Analysing the occurrences of *habit* in italics in the English–French part of the dictionary makes it possible to enrich the lexical entry described in the paper mentioned above. The monolingual lexical information for *habit–habitude* would be:

	habit	⟶	*habitude*
inchoative	acquire, contract, develop, form, get into, take to		prendre, contracter
terminative	break off, drop, conquer, forsake, get rid of, give up, outgrow, relinquish, shake off, slough off, throw off		abandonner, se débarrasser de, se défaire de, perdre, renoncer à, rompre avec, surmonter, vaincre
causative	infix		inculquer
causative + terminative	wean (from)		détacher (de), détourner (de).

Again, it should be noted that the aspectual values of support verbs can also be represented in terms of lexical functions: **Incep** parallels the inchoative value, **Cont** refers to the durative aspect and **Fin** refers to the end of a process/state (terminative) (see also Steele 1990, 47). Heid (1992) and Heylen *et al.* (1992) also show how lexical functions can be used in the construction of multilingual lexicons for machine translation.

It may be interesting to have a look at the data provided by the BBI Combinatory Dictionary of English (Benson *et al.* 1986). The entry for *habit* reads as follows:

habit *n.* ['custom'] ['usual manner'] 1. to acquire, develop, form a ~ 2. to make a ~ of smt. 3. to get into a ~ 4. to break a ~; to get out of a ~; (slang) to kick the ~ 5. to break smb. of a ~ 6. an annoying; bad; entrenched, ingrained; filthy; good; incurable; nasty; repulsive ~ 7. irregular; regular ~s 8. a ~ of (he has a bad ~ of interrupting people) 9. by force of ~ 10. out of ~ (I did it out of ~) ['costume'] 11. a monk's; nun's; riding ~

As can be noticed, this dictionary seeks to capture idiosyncratic collocations (support verbs, typical adjective-noun combinations, together with grammatical collocations). The main drawback, however, is that it does not attempt to do the semantic interpretation the authors describe in the introduction to the dictionary (the distinction between CA and EN collocations, for example). Of course, synonymous collocations that share a common semantic component are listed in a string (to *acquire*, *develop*, *form a habit*), but the reader has to discover the deep meaning of the collocation since no indication tells him/her that the verb has, say, an inchoative, a durative, a terminative or a causative meaning.

Conclusion

The aim of this paper was to show that computational dictionary analysis can complement corpus-based lexicography in a useful way, since commercial dictionaries embody a lot of useful information that can be put to good use in a lexicographer's workbench. As noted by Smadja (1991), co-occurrence knowledge rarely exists in compiled form. A notable exception is the BBI dictionary, but it is only available in print and the semantic interpretation described in the introduction has to be done by the user. The type of approach I sketch in this paper should be seen as a contribution to lexical acquisition in order to pave the way for better multilingual collocational dictionaries usable in language generation and in machine translation.

The ultimate aim is also to enrich the monolingual lexica by adding a collocational dimension to the classical description of lexical items. We have seen that doing so can be economic in a machine translation approach since it greatly reduces the size of the transfer lexicon. Various formalisms, ranging from Gross's lexicon-grammars to Mel'čuk's Meaning-Text Theory can then be used to exploit the type of data extracted from a bilingual dictionary such as the Robert & Collins.

References

Apresyan, Y.–Mel'čuk, I.–Žholkovsky, A. 1969. Semantics and lexicography: towards a new type of unilingual dictionary. In: Kiefer, F. (ed.): Studies in Syntax and Semantics, 1–33. Reidel, Dordrecht-Holland.

Atkins, B.T.–Duval, A. 1978. Collins & Robert: French–English English–French dictionary. Collins, London–Glasgow; Dictionnaires Le Robert, Paris.

Benson, M.–Benson, E.–Ilson, R. 1986. The BBI Combinatory Dictionary of English. John Benjamins Publishing, Amsterdam.

Boguraev, B. 1991. Building a lexicon: The contribution of computers. In: International Journal of Lexicography 4: 227–60.

Calzolari, N. 1989. Lexical databases and textual corpora: perspectives of integration for a lexical knowledge base. In: Proceedings of the First International Lexical Acquisition Workshop. Detroit.

Church, K.–Hanks, P. 1990. Word association norms, mutual information and lexicography. In: Computational Linguistics 16: 22–9.

Cruse, D.A. 1986. Lexical Semantics. Cambridge University Press, Cambridge.

Danlos, L.–Samvelian, P. 1992. Translation of the predicative element of a sentence: category switching, aspect and diathesis. In: TMI-92: Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation, 21–34. Montreal.

Fontenelle, Th. 1992a. Automatic extraction of lexical-semantic relations from dictionary definitions. In: EURALEX'90 Proceedings—Fourth International Congress, 89–103. Biblograf, Barcelona.

Fontenelle, Th. 1992b. Collocation acquisition from a corpus or from a dictionary: a comparison. In: Proceedings of the EURALEX Fifth International Congress, 221–28. Tampere, Finland.

Gross, M. 1981. Les bases empiriques de la notion de prédicat sémantique. In: Langages 63: 7–52.

Heid, U. 1992. Décrire les collocations: deux approches lexicographiques et leur application dans un outil informatisé. In: Terminologie et Traduction, n° 2–3, C.E.C., Luxembourg (Proceedings of the "Colloque Anniversaire de l'ETI", Genève, October 1991), 523–48.

Heylen, D.–Humphreys, L.–Warwick-Armstrong, S.–Calzolari, N.–Murison-Bowie, S. 1992. Collocations and the lexicalisation of semantic operations: lexical functions for multilingual lexicons. In: Haenelt–Wanner (eds): Proceedings of the International Workshop on The Meaning-Text Theory, Arbeitspapiere der GMD 671, Gesellschaft für Mathematik und Datenverarbeitung, 173–85. Darmstadt.

Machonis, P. 1991. The support verb 'make'. In Kiefer, F. (ed.): Computational Lexicography: Proceedings of the COMPLEX Conference, 141–53. Budapest, Research Institute for Linguistics, Hungarian Academy of Sciences, Budapest.

Mel'čuk, I.–Žholkovsky, A. 1988. The Explanatory Combinatorial Dictionary. In Evens, M. (ed.): Relational Models of the Lexicon. Cambridge University Press, Cambridge.

Mel'čuk, I. *et al.* 1984. Dictionnaire explicatif et combinatoire du français contemporain. Recherches Lexico-Sémantiques. Vol. I. Les Presses de l'Université de Montréal, Montréal.

Smadja, F. 1991. Macrocoding the lexicon with co-occurrence knowledge. In: Zernik, U. (ed.): Lexical Acquisition: Using On-Line Resources to Build a Lexicon. Lawrence Erlbaum Associates, Hillsdale, NJ.

Steele, J. (ed.). 1990. Meaning-Text Theory: Linguistics, Lexicography, and Implications. University of Ottawa Press, Ottawa.

Address of the author: Thierry Fontenelle
English Department
University of Liège
3 Place Cockerill
B–4000 Liège
Belgium

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 123-135 (1992-93)

UNE ANALYSE SÉMANTIQUE ET SYNTAXIQUE DES PHRASES À VERBES SUPPORTS DE L'ALLEMAND ET DU FRANÇAIS*

KATHARINA GREWE

1. Introduction

Cet article se propose d'étudier des constructions nominales prédicatives comme

(a.1) *eine Frage stellen*
(b.1) *in Erwägung ziehen*
(c.1) *in Wut versetzen*

Il est de tradition dans la philologie allemande de désigner ces syntagmes par «Funktionsverbfügung» (Heringer 1968), «Funktionsverbgefüge» (Engelen 1968), «Funktionsverbformeln» (Erben [11]1972), «Streckformen» (Schmidt 1968), etc. Dans ces phrases, les verbes sont dits «Streckverben» (Reiners 1943; Häusermann 1977) et «Funktionsverben» (von Polenz 1963). Même l'élément nominal ne connaît pas un terme unique: «Gefügenomen» (Engel 1988), «Funktionsnomen» (Herrlitz 1973), etc.

En français, de telles expressions sont aussi nombreuses:

(a.2) *poser une question*
(b.2) *prendre en considération*
(c.2) *mettre en rage*

Comme le «discours répété» en général, les FVG du français sont un terrain peu exploré. On manque de terminologie suffisamment large et universellement acceptée. On a généralement considéré les phrases à verbes supports comme un type des unités phraséologiques (Bally 1905). On compte, il est vrai,

* Ce travail s'est effectué dans le cadre du projet sur la structure sémantico-syntaxique d'un dictionnaire, de l'Université de Bochum, Sprachwissenschaftliches Institut (Prof. Dr. Helmut Schnelle) et dans le cadre d'une épreuve écrite d'examen à l'Université de Bochum en 1992 sous le titre «Funktionsverbgefüge im Französischen und Deutschen».

ces unités complexes appelées «série verbale», «périphrase verbale», «locution verbale», «tour verbal» et «phrase à verbe support» parmi le matériel linguistique. Ce qu'on appelle FV en allemand correspond à la notion française de «verbe pauvre» (Barth 1961), «verbe signe» (Grevisse [11]1980), «verbe opérateur» (Giry-Schneider 1978a, b) et «verbe support» (Daladier 1978). Le phénomène lui-même est très complexe comme les désignations l'indiquent. Quels sont les traits caractéristiques qui distinguent le FVG (a) *mettre en fureur* des syntagmes (b) *mettre à l'ombre* et (c) *mettre (son argent) à la caisse d'épargne* dans les phrases suivantes?

(a) *Luc a mis Jean en fureur*
(b) *Le cambrioleur a été mis à l'ombre*
(c) *Il met son argent à la caisse d'épargne*

L'existence de cette relation en allemand apparaît avec les expressions suivantes du verbe *halten* comme le FVG (a) *ein Versprechen halten* et les constructions (b) *den Mund halten* et (c) *einen Stift halten*. Ces exemples font apparaître que les FVG présentent des structures syntaxiques qui s'appliquent non seulement à des syntagmes libres mais encore à quelques unités phraséologiques.[1] Par conséquent ce qui nous importe ici, c'est la question du statut grammatical à attribuer aux FVG. Les FVG prennent pour ainsi dire une position intermédiaire: d'un côté ils ne sont pas des syntagmes libres et d'un autre côté ils ne sont pas non plus des locutions idiomatiques. Les FVG font partie des cas limites de la linguistique.[2]

2. «Funktionsverbgefüge» et unités phraséologiques

Ainsi, la définition traditionelle d'une unité phraséologique qui est toujours la plus répandue, tient compte de l'irrégularité sémantique, de quelques défauts transformationnels, de la stabilité sémantique-syntaxique, de la reproduction, de l'idiomaticité, etc. Ces critères sont partiellement les mêmes pour les FVG. C'est pour cela qu'ils ont seulement de l'importance pour la différence entre des FVG et des mots libres.

[1] «L'unité phraséologique» doit être comprise comme une notion générale pour tous les types de phraséologismes.

[2] «It is possible that we are approaching here the fringe of marginal cases, to be expected in a system as complex as a natural language, where significant systematization is just not possible» (Chomsky 1965, 192).

Les FVG ne sont pas d'abord produits comme des syntagmes libres dans la parole, mais ils sont classés comme les autres unités phraséologiques du système de la langue. Ils sont entièrement reproduits dans la parole.

Le sens des expressions idiomatiques n'est pas analysable à partir de la signification des éléments constitutifs. La relation entre les signifiés des constituants et la signification de la locution globale est irrégulière. Par exemple *auf die Folter spannen* dans le sens de *jemanden auf etwas gespannt machen, jemandes Neugierde absichtlich nicht befriedigen*, le processus d'idiomatisation est total. Aucun des constituants de cette tournure ne conserve sa signification lexicale.

Les phrases figées sont ambigues et l'interprétation littérale (voir les exemples (a.1; a.2)) est toujours possible. Voici quelques exemples:

(a.1) *mettre qqch dans sa poche*, *montrer les dents à qqn*, *jeter qqch à la tête de qqn*
(a.2) *etw in die Tasche stecken*, *jmdm die Zähne zeigen*, *jmdm etw an den Kopf werfen*
(b.1) *mettre qqn dans sa poche* (familier), *montrer les dents à qqn*, *jeter qqch à la tête de qqn*
(b.2) *jmdn in die Tasche/Sack stecken* (familier), *jmdm die Zähne zeigen*, *jmdm etw an den Kopf werfen*

Par contre, le sens d'un FVG résulte de chaque constituant du syntagme. En général, les FVG sont désignés comme unités phraséologiques parce qu'ils sont fixes et forment une unité sémantique. Les lexèmes dans un FVG ne diffèrent pas d'une norme grammaticale lexique. En considération de la subcatégorisation stricte, la plupart des expressions idiomatiques ne sont pas bien formées: *jmdm eins aufs Dach geben*, *faire semblant.*

Dans les expressions idiomatiques, les substantifs sont normalement définis comme sémantiques «concrets» (a), ou dans les FVG, ils restent abstraits (b):

(a) *tourner la tête à qqn*,*marcher sur les talons de qqn*, *jmdm den Kopf verdrehen*, *jmdm (dicht) auf den Fersen sein*

(b) *faire un aveu*, *ein Geständnis ablegen*

3. Critères de classification

Nous allons maintenant ébaucher les propriétés les plus importantes qui ont permis d'isoler les constructions à verbes supports des constructions à verbes pleins et également des constructions à valeur idiomatique.

Les FVG sont des phrases formées d'un FV et un complément d'objet direct ou un complément prépositionnel. Tout d'abord, on va classer les locutions verbales en question selon leur construction superficielle:

- les FVG du type I

  Les FVG du type I (FV+PP) peuvent être classés en trois sous-groupes:

  1. Prép. + Dét. zéro + FN + FV[3]

  (a) *mettre à exécution*, *entrer en colère*, *prendre en aversion*, *être hors de doute*, *tomber en putréfaction*

  (b) *in Haft nehmen*, *in Wut bringen*, *in Betrieb setzen*, *in Ordnung bringen*, *in Betracht ziehen*, *in Erscheinung treten*

  2. Prép. + Dét. indéf. + FN + FV

  (a) *entrer dans une colère terrible*, *tomber dans une cruelle défaveur*, *entrer dans une fureur*

  (b) *in ein Gespräch eintreten*, *in eine Panik geraten*

  3. Prép. + Dét. déf./Dét. poss. + FN + FV

  (a) *mettre dans l'embarras*, *mettre à l'épreuve*, *mettre à la raison*, *mettre à la disposition*

  (b) *in den Ruin treiben*, *in seinen Besitz nehmen*, *auf der Flucht sein*, *zur Verarbeitung kommen*, *zur Durchführung gelangen*, *zum Lachen bringen*

- les FVG du type II

  Les FVG du type II (FV+NP) sont à subdiviser selon la structure superficielle en trois sous-groupes:

[3] Le schéma tient compte seulement de la composition d'un FVG et pas de la suite des constituants d'une langue spécifique.

1. Dét. zéro + FN + FV

(a) *faire grève, prendre connaissance, rendre hommage, mettre fin, faire attention, avoir de l'admiration*

(b) *Bezug nehmen, Unterstützung geben/finden, Dankbarkeit zeigen, Hilfe leisten, Verwendung finden*

2. Dét. indéf. + FN + FV

(a) *faire un plongeon, faire une promenade, faire un sourire, recevoir une gifle, tirer une conclusion*

(b) *eine Wendung nehmen, eine Untersuchung vornehmen, eine Erklärung abgeben, einen Schrei ausstoßen*

3. Dét. déf./Dét. poss. + FN + FV

(a) *faire l'examen, faire le récit, faire l'achat, prendre la fuite, prendre ses distances*

(b) *der Meinung sein, der Ansicht sein, den Beweis führen, die Flucht ergreifen, den Schluß ziehen*

Mais cette représentation syntaxico-morphologique des FVG ne contribue pas beaucoup à différencier les FVG des locutions idiomatiques ainsi que des unités syntaxiques dites libres.

Le FV n'est qu'un outil morphologique et syntaxique qui sert pratiquement à conjuguer le FN. Ces verbes sont des marques de temps, de personne et de nombre. Il n'y a pas de verbes quelconques qui peuvent avoir la fonction d'un FV dans un FVG. Souvent il s'agit de verbes polysèmes, c'est-à-dire de verbes avec plus de significations. En français, les verbes *donner*, *pousser*, *faire*, *prendre*, *rendre*, *avoir*, *être*, *aller*, *mettre*, *entrer*, *tomber*, *tirer* et *demander* sont les verbes les plus utilisés de la catégorie FV. En allemand, les verbes *bringen*, *kommen*, *sein*, *stehen*, *geraten*, *setzen*, *stellen*, *halten*, *nehmen*, *bleiben* et *haben* sont les FV les plus utilisés.

Avec ces verbes, il y a des substantifs qui, malgré leur place et leur forme apparente, ne sont pas des arguments du verbe principal. Du point de vue sémantique, un nom (FN) en relation avec un FV est porteur du sens lexical de l'expression. Le prédicat représenté par le nom prédicatif désigne des actions, des activités, des événements, des états, etc. Les FN sont des substantifs abstraits, dérivés d'un verbe ou d'un adjectif. En illustration les exemples suivants:

(a.1) *prendre la fuite*
(a.2) *entrer en balance*
(b.1) *être en colère*
(a.3) *zum Verschwinden bringen*
(a.4) *in Schwung bringen*
(b.2) *in Verlegenheit bringen*

Comme le montrent les exemples ci-dessus, les FN de ces FVG décrivent presque toujours des actions et des situations ((a.1)–(a.4)). Seulement deux substantifs abstraits démontrent un état ou une propriété ((b.1); (b.2)).

En comparaison à son verbe de base ou à son adjectif de base, le FVG se différencie souvent, soit dans son aspect, soit dans la signification du verbe principal:

- les FVG «sémantiques» sont différents des lexèmes principaux
  (a) *mettre en danger (être dangereux), faire un récit (réciter)*
  (b) *in Verlegenheit bringen (verlegen sein), zu Ansehen bringen (ansehen)*

- les FVG «simples» comme synonymes des lexèmes principaux
  (a) *mettre en doute (douter de), donner une réponse (répondre à)*
  (b) *in Zweifel ziehen ((be)zweifeln), Antwort geben (antworten)*

4. Données de combinatoire lexicale

Une particularité des FVG est sa tendance de former une autre expression avec un FV comme élement stable. Le FV peut se construire avec d'autres FN de séries d'oppositions plus ou moins complètes:

prendre	*treffen*
prendre une décision	*eine Entscheidung treffen*
prendre un arrangement	*eine Vereinbarung treffen*
prendre des dispositions	*Vorkehrungen treffen*
prendre	*ergreifen*
prendre la fuite	*die Flucht ergreifen*
prendre l'initiative de	*die Initiative ergreifen*
prendre des mesures	*Maßnahmen ergreifen/treffen*

prendre possession de	*Besitz ergreifen von*
prendre des précautions	*Vorsichtsmaßnahmen ergreifen/treffen*
prendre la parole	*das Wort ergreifen*

Mais il y a des FVG qui peuvent être construits par le FV comme élément remplaçable et par le FN comme élément stable:

(a.1) *être, mettre, entrer en action*
(a.2) *être, mettre, maintenir en marche*
(b.1) *in Bewegung sein, bringen, kommen*
(b.2) *in Gang sein, setzen, halten*

Les FV et les FN également ont des extensions lexicales. On ne discutera pas ici de la question beaucoup trop générale de la paraphrase et de la synonymie. Les significations ne diffèrent pas beaucoup:

(a.1) *donner/intimer l'ordre*
(a.2) *donner/fournir une réponse*
(b.1) *Befehl geben/erteilen*
(b.2) *eine Antwort geben/erteilen*

Les FVG ne sont pas une catégorie homogène. Ils se différencient en divers degrés de la stabilité de leurs composants. Les FVG sont plus ou moins des expressions figées, c'est-à-dire qu'il existe des expressions plus lexicalisées (a) et aussi des expressions moins stables (b):

(a.1) *entrer en ligne de compte, tenir conseil*
(b.1) *mettre à la disposition, tomber dans la misère*
(a.2) *in Frage kommen, zu Rate gehen*
(b.2) *zur Verfügung stellen, in Not geraten*

5. Données sémantiques

5.1. «Funktionsverb» et verbe simple

En fin de compte, ce serait sans doute l'étude sémantique qui devrait fournir le critère décisif. Il y a en effet un ensemble de propriétés sémantiques qui différencient les FVG.

Chaque FVG forme une unité sémantique indépendante qui sert de verbe dans la phrase. Certains noms forment avec des verbes tels que *faire*, *avoir*, etc. des locutions verbales équivalentes à des verbes simples. La substitution d'un FVG à un verbe de même radical prouve sa cohésion sémantique. Ainsi, le FVG est utilisé comme synonyme du verbe de base. Considérons la périphrase verbale *Max fait des compliments à Léa*; elle est synonyme de la phrase à verbe *Max complimente Léa.*

(a.1) *prendre en considération* et *considérer*
(b.1) *in Erwägung ziehen* et *erwägen*
(a.2) *poser une question* et *questionner*
(b.2) *eine Frage stellen* et *fragen*

- les FVG sont remplaçables par le verbe principal sans éprouver une perte d'information:

  (a.1) *porter un jugement sur qqch/qqn (juger)*
  porter une (grande) admiration (admirer)

  (a.2) *prendre en haine (hair)*

  (a.3) *Nachricht geben (benachrichtigen)*

- les FVG sont normalement les seules possibilités d'exprimer l'action verbale:

  1. les FVG n'ont pas d'équivalents synthétiques

     (a.1) *mettre au courant (informer)*

     (a.2) *mettre en danger (compromettre)*

     (b.1) *in Kenntnis setzen (mitteilen)*

  2. a) le verbe de base est désuet ou inutilisé

     entrer en colère (vieilli: *se colérer*)

     b) le FVG est désuet ou inutilisé

     (vieilli) *mettre en oubli (oublier)*
     (veraltet) *in Harnisch bringen (wütend machen)*

  3. les FVG expriment des significations essentielles contrairement à leur verbe principal:

     a) le FVG a une signification spéciale

(a) *mettre en observation (observer)*
(b) *unter Beobachtung stellen (beobachten)*

b) le FVG a une signification générale

entrer dans le sommeil (sommeiller)

Une indication poussée de l'unité sémantique d'un FVG est la nominalisation de l'expression. Mais cette opération n'est pas toujours usuelle:

(a.1) *la mise en service*
(b.1) *die Inbetriebnahme*
(a.2) *la mise en état*
(b.2) *die Instandsetzung*

Beaucoup de FV (a.1; b.1) signalent une signification à voix passive (a.2; b.2):

(a.1) *qqch entre en accomplissement*
(a.2) *qqch est accompli*
(b.1) *etw kommt zur Vollendung*
(b.2) *etw wird vollendet*

5.2.1. Phases du procès et causativité

La principale fonction sémantique réside dans la précision ou la modification d'une information sémantique spécifique exprimée par le FV comme les FVG in *Gang sein/être en marche*, *in Gang kommen/entrer en marche* et *in Gang bringen/mettre en marche* le montrent. Ces modifications sont nommées en philologie par le terme «Aktionsart» et elles se manifestent en signes comme durée, finalité, causativité et leurs combinaisons. La notion allemande d'"Aktionsart» est difficile à traduire en français. Elle a été confondue avec celle d'aspect. Pour éviter les confusions des termes d'"Aktionsart» et d'aspect, nous avons préféré la notion allemande dans cet article.

Les FVG se répartissent en trois sous-groupes, selon qu'ils expriment (a) l'entrée dans un état ou le début d'une action, (b) l'action en déroulement et (c) le changement d'état provoqué. Le verbe principal (d) est déterminé par une réalisation certaine d'un procès.

(a) *entrer en marche, in Bewegung kommen*
(b) *être en marche, in Bewegung sein*

(c) *mettre en marche, in Bewegung setzen*
(d) *marcher, (sich) bewegen*

Dans l'échange d'un FV avec une variante opposée, la signification varie:

(a.1) *avoir de l'aversion pour qqn, avoir un certain balancement*
(a.2) *prendre en aversion, prendre un certain balancement*
(a.3) *garder un certain balancement*
(a.4) *perdre (tout + son) balancement*
(b.1) *avoir peur*
(b.2) *prendre peur*
(b.3) *faire peur*
(c.1) *in Bewegung sein, in Bewegung bleiben*
(c.2) *in Bewegung setzen, in Bewegung bringen, in Bewegung versetzen*
(c.3) *in Bewegung kommen, in Bewegung geraten*

Elles expriment alors les différentes phases d'un procès:

- la durativité
- la transformativité

La durativité démontre le déroulement d'un procès acutel. La durativité est représentée par des verbes tels que *être*, *avoir*, *rester*, *demeurer*, *haben*, *sein* et *bleiben*.

(a) *rester en ordre, être en marche, être en colère, être à la disposition, avoir à sa disposition*

(b) *in Ordnung bleiben, in Gang sein, in Wut sein, zur Verfügung stehen, zur Verfügung haben*

Le mode d'action du type transformatif exprime le pasage d'un procès à un autre ou limite le déroulement du procès. La transformativité se divise en deux *Aktionsarten*:

- l'inchoativité[4]

[4] Appelé aussi «ingressif».

Un verbe est alors qualifié d'inchoatif si son début est marqué (entrée en l'état). La phase inchoative est identifiée par les verbes suivants: *tomber*, *entrer*, *kommen* et *geraten*.

(a) *prendre peur*, *entrer en relation*, *prendre connaissance*, *entrer en balance*, *entrer dans une colère noire*

(b) *in Angst geraten*, *in Verbindung treten*, *Kenntnis nehmen*, *in blinde Wut geraten*

- l'égressivité[5]

Le mode d'action du type égressif marque la fin de l'action ou le résultat d'un procès:

(a) *eine Verbesserung herbeiführen*, *eine Untersuchung durchführen*

(b) *apporter une amélioration*, *réaliser une étude*

Les FVG servent aussi à exprimer un fait de causativité. La causativité, contrairement à ce qui est écrit dans de nombreux travaux sur les «Aktionsarten», n'est pas une «Aktionsart» si l'on entend par ce terme le mode temporei de déroulement du procès.

(a) *mettre à la disposition, mettre en mouvement*

(b) *zur Verfügung stellen, in Bewegung bringen*

Elle correspond le plus souvent à l'introduction d'un argument supplémentaire de la phrase simple de départ (von Polenz 1987; Persson 1975):

(a.1) *Le moteur marche*
(a.2) *Il met le moteur en marche*
(b.1) *Der Motor läuft*
(b.2) *Er bringt den Motor in Gang*

La causativité peut être assimilée à un trait facultatif pouvant se combiner avec chacune des «Aktionsarten». Par exemple les verbes causatifs *mettre* et *bringen* du type I sont très productifs en français et en allemand:

[5] Appelé aussi «résultatif».

(a.1) *mettre en colère*
(a.2) *mettre en sûreté*
(b.1) *in Wut bringen*
(b.2) *in Sicherheit bringen*

L'application de l'opérateur causatif ne s'appliquent qu'aux certains verbes. Giry-Schneider (1986, 55–6) a ainsi montré qu'il y a une relation entre les phrases à *avoir* et *il y a* ou à *faire* et *donner*:

(a.1) *Max a faim*
(a.2) *Ceci donne faim à Max*
(b.1) *Paul a eu un choc*
(b.2) *Cette nouvelle a fait un choc à Paul*
(c.1) *Max hat Hunger*
(c.2) *Das macht ihm Hunger → Das macht ihn hungrig*
(d.1) *Paul hat einen Schock (gehabt)*
(d.2) *Diese Nachricht hat bei Paul einen Schock verursacht/ausgelöst*

Bibliographie

Bally, Ch. 1905. Précis de stylistique. Esquisse d'une méthode fondée sur l'étude du français moderne. A. Eggimann, Genève.

Barth, G. 1961. Recherches sur la fréquence et la valeur des parties du discours en français, en anglais et en espagnol. Didier, Paris.

Chomsky, N. 1965. Aspects of the Theory of Syntax. MIT Press, Cambridge, MA.

Daladier, A. 1978. Problèmes d'analyse d'un type de nominalisation en français et de certains groupes nominaux complexes. Thèse de doctorat de 3e cycle. L.A.D.L. Paris.

Engel, U. 1988. Deutsche Grammatik. Groos, Heidelberg.

Engelen, B. 1968. Zum System der Funktionsverbgefüge. In: Wirkendes Wort 18: 289–303.

Erben, J. [11]1972. Deutsche Grammatik. Ein Abriß. Hueber, München.

Giry-Schneider, J. 1978a. Les nominalisations en français. L'opérateur «faire» dans le lexique. (Langue et Cultures 9). Droz, Genève.

Giry-Schneider, J. 1978b. Interprétation aspectuelle des construnctions verbales à double analyse. In: Linguisticae Investigationes II 1: 23–53.

Giry-Schneider, J. 1986. Les noms construits avec *faire*: compléments ou prédicats? In: Langue Française 69: 49–63.

Grevisse, M. [11]1980. Le bon usage. Grammaire française avec des remarques sur la langue française d'aujourd'hui. Duculot, Paris; Gembloux.

Grewe, K. 1992. Funktionsverbgefüge im Französischen und Deutschen. Magisterarbeit am Institut für Romanische Philologie. Ruhr-Universität Bochum.

Gross, G.–Vivès, R. (eds). 1986. Les constructions nominales et l'élaboration d'un lexique-grammaire. In: Langue Française 69: 5–27.

Gross, M. 1981. Les bases empiriques de la notion de prédicat sémantique. In: Langages 63: 7–52.

Häusermann, J. 1977. Phraseologie. Hauptprobleme der deutschen Phraseologie auf der Basis sowjetischer Forschungsergebnisse. (Linguistische Arbeiten 47). Niemeyer, Tübingen.

Heringer, H.-J. 1968. Die Opposition von „kommen" und „bringen" als Funktionsverben. Untersuchungen zur grammatischen Wertigkeit und Aktionsart. (Sprache der Gegenwart 3). Pädagogischer Verlag Schwann, Düsseldorf.

Herrlitz, W. 1973. Funktionsverbgefüge vom Typ „in Erfahrung bringen". Ein Beitrag zur generativ-transformationellen Grammatik des Deutschen. (Linguistische Arbeiten 1). Niemeyer: Tübingen.

De Negroni-Peyre, D. 1978. Nominalisation par «être en» et réflexivation. In: Linguisticae Investigationes II 1: 127–64.

Persson, I. 1975. Das System der kausativen Funktionsverbgefüge. Eine semantisch-syntaktische Analyse einiger verwandter Konstruktionen. (Lunder germanistische Forschungen 42). Gleerup, Lund.

Polenz, P. von 1963. Funktionsverben im heutigen Deutsch. Sprache in der rationalisierten Welt. (Beiheft zur Zeitschrift Wirkendes Wort 5). Düsseldorf.

Polenz, P. von 1987. Funktionsverben, Funktionsverbgefüge und Verwandtes. Vorschläge zur satzsemantischen Lexikographie. In: Zeitschrift für Germanistische Linguistik 15: 169–89.

Reiners, L. 1943. Stilkunst. Ein Lehrbuch deutscher Prosa. Becksche Verlagsbuchhandlung, München.

Schmidt, V. 1968. Die Streckformen des deutschen Verbums. Substantivisch-verbale Wortverbindungen in publizistischen Texten der Jahre 1948–1967. VEB Niemeyer, Halle/Saale.

L'adresse de l'auteur: Katharine Grewe
Sprachwissenschaftliches Institut
Ruhr-Universität Bochum
Universitätstr. 150
Postfach 10 21 48
D-44780 Bochum
Fax: 49-234-70 49-137
Email: grewe @ linguistics.ruhr-uni-bochum.de

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 137-157 (1992-93)

EXTRACTING LINGUISTIC INFORMATION FROM MACHINE-READABLE VERSIONS OF TRADITIONAL DICTIONARIES: A METALEXICOGRAPHIC METHOD AND SOME TOOLS

ULRICH HEID-MATTHIAS HEYN-OLIVER CHRIST

1. Goals and framework

1.1. The problem

Most high-level Natural Language Processing (NLP) systems need large amounts of detailed linguistic information, and since dictionary construction is a time consuming activity, it is important to investigate possibilities of reusing electronically available versions of "traditional" published paper dictionaries. In such a reuse situation, a paper dictionary would serve as a source of information, and the dictionary of the intended NLP system or the lexical requirements of the theoretical approach underlying the system would constitute the "expectation horizon" of the intended application of lexical knowledge to be filled entirely or partly with material from the source. In Fig. 1 we show the working steps towards the reuse of machine readable versions of published dictionaries. The practical work necessary to extract linguistic information from an electronic version of a traditional dictionary involves a number of conceptually different steps which are often performed together. These steps involve the *reformatting* of the available data and the extraction of relevant elements from there, wherever possible and useful. To this end, we have to gain an understanding of the presentational devices used in the dictionary, by *reinterpreting* the data. These processes, of reinterpretation and reformatting, often performed at a time, can be a very time-consuming, iterative and error-prone enterprise. We thus consider it useful to do first "explorative" experiments before programming any computational tool for the reuse of a particular dictionary. The present paper is about such an "explorative" study and about a general methodology of such work.

Such a preliminary "explorative" study is most necessary if a *"traditional"* dictionary is considered as a potentially reusable source of linguistic information. We call "traditional dictionaries" those published dictionaries which have been produced according to the traditional lexicographic working procedures, without use of a computational system to enhance consistency, to facilitate the

Akadémiai Kiadó, Budapest

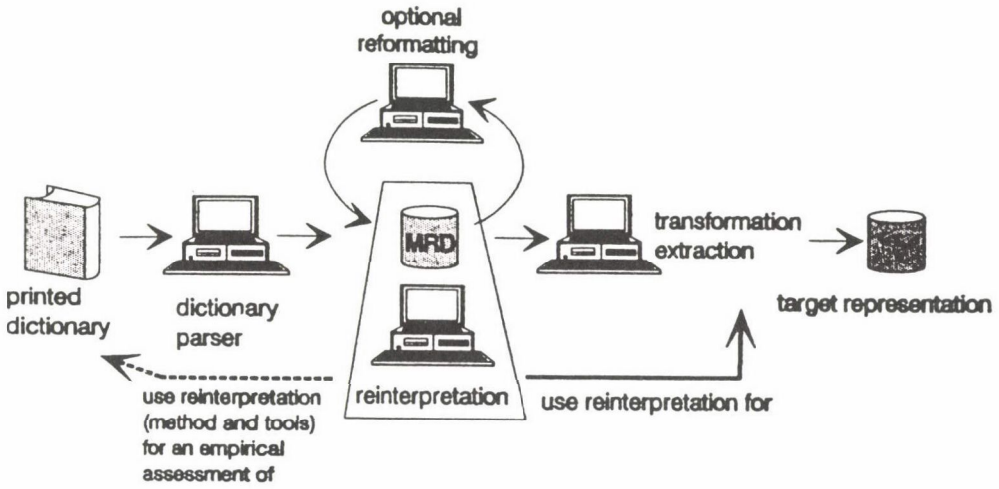


Fig. 1
Working steps towards the reuse of MRDSs

interaction among the authors of dictionary articles or to check whether the authors of articles stick to the guidelines given in the lexicographic instruction manual underlying their work. The third edition of the *Oxford Advanced Learner's Dictionary of Current English* is an example of such a "traditional dictionary". Since it is easily accessible for research purposes, we have used it for the work described in the present article.[1] However, for the argument of this paper, the use of this dictionary has much of an exemplary case-study; we do not wish to discuss in all detail the results obtained with this particular dictionary, but we are interested in the general types of problems posed by the reuse of traditional dictionaries and in methods and tools for the exploration of such dictionaries.

The reuse of a traditional dictionary for the construction of an NLP lexicon involves, as we call it, *reformatting* and *reinterpretation*. By *reformatting*, we understand the process of translation from one representation format into another, say from a typesetting format to a representation of the dictionary as

[1] We would like to thank Oxford University Press for making available the electronic edition (OALD 3e) to us for research purposes. In the meantime, after we had finished our studies of the OALD 3e, the fourth edition of the OALD, completely revised and reorganized, has become available in book and machine readable form. We could not yet analyze the machine readable version of OALD 4, for time reasons.

a text file with a markup.[2] In the case of OALD we only had to reformat from an SGML-like format to a notation as lists of LISP. This because conventional SGML-based tools could not be applied, since the OALD 3e does not come with a document type definition for the tools to operate on. The OALD typesetting tapes have been parsed by Oxford University Press, and this passing procedure has produced a marked-up text file. The markup used is inspired by the SGML conventions, but since the parsing in essence operated on the use of typesetting conventions in the dictionary, without tempting to reconstruct any particular article syntax, no DTD was produced along with the marked-up text. The *reformatting* process as such is nothing but an exchange of the encoding: it does not touch upon the signification of the dictionary text.

Reinterpretation, however, is the process of linguistic and (meta-)lexicographic analysis of the contents of the dictionary text. Metalexicographers sometimes distinguish between the descriptive and the presentational side of lexicographic work. They thereby capture the fact that dictionary writing consists, roughly speaking, of two phases: a first one, *description*, where the dictionary author gathers material, structures and organizes it and describes the evidence he has before him, according to a given model; this model may be more or less formal(izable), of course; most often, it is encoded in the instruction manual. The second phase, *presentation*, aims at the writing of a dictionary article, which is a specific type of text, governed by rules; these rules ideally constitute the "syntax" or the "grammar" of the dictionary articles. Texts of dictionary articles are known to make use of numerous *presentational devices*, such as typographical change, abbreviations, punctuation, separator signs, sequencing and hierarchical organization of text elements, and so on. All these devices are meaningful, and so is the "wording" of the dictionary text: the dictionary author uses a number of devices to convey his descriptive intuition about the facts he has observed in the material he has gathered and analyzed. The task of reinterpretation is to understand the descriptive intuition of the dictionary author and to reconstruct, at least partially, the fragment of the linguistic facts of a language covered by his description. The reuse of a dictionary is the more easily and efficiently achievable the better we manage to relate elements of the textual presentation we find in dictionaries with observable linguistic phenomena or with our own way of formally describing these phenomena and of representing these formal descriptions. If some dictionary marks "transitive verbs" with a code "vt", it may be too naive

[2] For example, this had to be done with LDOCE within the work done by the Cambridge group. See e.g. Alshawi *et al.* (1989).

to hope that we can safely assume that a formal grammar underlying an NLP system would necessarily classify the same set of lexical elements as "transitive verbs". But through reinterpretation, we want to get an idea of the facts which have been considered by the dictionary authors to be relevant for his articles. Maybe he makes a number of distinctions which we would collapse into one descriptive class; or he does not distinguish a number of phenomena which we need to distinguish in our NLP application.

The goal of a reuse exercise maybe to explore the dictionary as a whole and to reuse all information available, or at least as much as possible; we call this type of reuse activity *transformation*. Alternatively, only part of the information contained in a dictionary may be looked up automatically, e.g. to find evidence for a linguistic hypothesis or for a classification of certain types of facts; we call this type of reuse activity *extraction*. Extraction work is useful once we know which facts are most reliably and conveniently described in a given dictionary: we will then try to extract those.

1.2. The task

Our work is based on the *Oxford Advanced Learner's Dictionary*, OALD. The concrete task of reformatting and reinterpretation of the information contained in the OALD falls into two subtasks, each of which requires a certain set of methods and tools. The first subtask is to describe in detail the internal text structure of the different types of articles found in this dictionary. On the basis of the results of this, we can attempt the second subtask which is to investigate the availability, statistical relevance and descriptive adequacy of (certain types of) the linguistic descriptions presented in the dictionary.

An analysis of the internal structure of the articles is indispensable for a detailed and concrete enough description of the interdependencies of certain pieces of the article text; knowledge about these interdependencies is in turn needed for the formulation of extraction and transformation statements for a dictionary parser. The sequencing of different types of indications in a dictionary article text is not arbitrary, but may depend on a number of parameters, such as the type of the article, the presence or absence of certain other types of indications, etc. To be able to find the highest possible number of occurrences of a certain type of indication, we have to know about the places within article structures, in which the respective indication can appear. The more we know about article structure, the higher the percentage of extractable indications with respect to those present in the dictionary at all. Knowledge about dictionary structure helps thus enhance the "recall" of the tools to be built for extraction or transformation. In OALD 3, we have "recursive" article

structures: an article may have sublemmas (e.g. morphologically related with the main lemma, such as derivatives and compounds of the lemma), and the "subarticles" for such sublemmas follow in part the rules for the syntax of main articles. In such case, knowledge about article structure is necessary to be able to correctly relate lemmatic and sublemmatic indications respectively.

Knowledge about accessibility, statistical relevance and descriptive adequacy of the actual linguistic description available in the dictionary is a basis for decisions about (selective) extraction of certain types of information: it is important to have an idea of the quantity (statistical relevance) and the quality (descriptive adequacy) of the potential extraction or transformation results. Furthermore, we must know in detail about the interrelationship between the phenomena to be described and the descriptive devices used. Ideally, we expect that the lexicographer uses exactly one descriptive device for exactly one type of phenomena which needs to be described. However, often, such one-to-one relations are not consistently found throughout the dictionary. It happens that similar or identical facts are described by use of different descriptive devices (which then are "synonymous") or that one device is to be interpreted differently, depending on the context in which it appears. So, besides the place in the articles, the quantitative importance and the linguistic adequacy and reliability of indications given in a dictionary, we are interested in a picture of the devices used to convey descriptive results about lexical material.

The two subtasks of reuse actions depend on each other. Any assessment of the accessibility of a given piece of information heavily depends on knowledge about the internal structure of the articles. Consequently, first article structure needs to be analyzed, before individual types of indications can be assessed.

1.3. The approach, methods and tools

The work on the reinterpretation and reformatting of the OALD 3e combines *methods* from metalexicography, descriptive and theoretical linguistics with *tools* from "computational lexicographic practice": to arrive at describing in sufficient detail the internal structure of the articles of the OALD 3e, we have adapted the metalexicographic dictionary description method of Wiegand (1991).[3] A number of computational *tools* have been developed to check both dictionary article structures (along the lines of our modification of Wiegand's models) and the statistic relevance and the consistency of the linguistic descriptions presented in the OALD. This combination of a method for dictionary description with the tools supporting individual search and query tasks

[3] The most recent comprehensive account of Wiegand's methods for the description of dictionaries as texts can be found in Wiegand (1991).

does not lead to the formulation of a full dictionary entry parser, but it could serve as the input rule set for such a tool; the resulting set of detailed descriptive statements about the textual structure of the dictionary articles, as well as about the properties of the presentational devices used in the dictionary, would give, together, a sophisticated transformation routine, allowing to re-represent the OALD 3e in a well-defined homogeneous format.[4] Our work differs from that of researchers who would use the DTD of dictionary as a starting point for their reinterpretation exercise: in such case, they would have a pre-established guide-line which would allow them to determine precisely the place within dictionary articles where to look for certain types of information. Our task and approach are more comparable with the work by Bläsi and Koch (in this volume) who use a metalexicographic model of dictionary text structure to parse a given dictionary. They base their work on the same dictionary description method as we do: the work by Wiegand which gives the most detailed account of text structures of dictionary articles we know of. Bläsi and Koch implement a number of models of article structure and parse those articles of the Duden dictionary which follow the article, is to got a picture of the different models of article structure we would need for a (more or less complete) parse of the OALD 3e of results such a parsing of the dictionary would produce.

In the following section, we go through the most frequent types of situations occurring in the reinterpretation and reformatting of a traditional dictionary in view of its reuse in an NLP system. For each type of situation we indicate the problems to be solved and illustrate them with examples from the OALD 3e, then indicate the methods underlying the reinterpretation work and finally describe the functionality of the tools used to carry out the reinterpretation and reformatting exercise.

2. Applying the methods and tools to concrete problems

Extraction and transformation routines for traditional dictionaries are usually very complex and highly specialized, because they have to keep track of numerous anomalies of dictionary articles. Such anomalies often have to do with the hierarchical structure of dictionary articles, with missing (but "expected") information, implicit statements, polyfunctionality or synonymy of value names

[4] We have carried out tentative transformations of the dictionary; one into an attribute-value notation as supported by unification grammars (represented in CLOS, Common Lisp Object System), and one into a marked-up text file format supported by a commercial look-up tool (MultiTerm). The first transformation did, however, not use the full set of the results of the structural and linguistic analysis. Its aim was to demonstrate the feasibility of such a transformation as such; cf. Christ (1990).

or with idiosyncratic descriptions without statistical significance and with a problematic status within the descriptive work as a whole. Dictionary parsing presupposes ideally there to be a "grammar" for the textual structure of the dictionary articles (i.e. the order of functionally different text elements, interdependencies of text elements, etc. This "grammar" can for example be expressed by a document type definition (DTD) of SGML). With traditional dictionaries, such a grammar often lacks or is not very detailed.[5]

2.1. Article Structure: relating parts of article texts

Rules governing text structure and interdependencies between pieces of the article text differ from dictionary to dictionary and have thus to be derived by induction, from the dictionary text. If these rules for article structures are not detailed enough, the dictionary parser will run into problems. Most often, a dictionary is not just a collection of many structurally identical entries, but makes use of a number of article-structural types.[6] For modularity reasons, it makes sense to have a library of descriptions and rules for well-formed article structure types and then to add a particular treatment of "exceptions" where appropriate. For setting up these rules we have based our work on Wiegand's methods for the analysis of dictionary articles.

2.1.1. Some examples

Dictionary articles consist of numerous text elements which each have different functions; these text elements are sometimes called *items*; Wiegand's original

[5] Instruction manuals of dictionaries are a (more or less formal) description of the grammar of dictionary text. But these are usually not available for computational lexicographic or metalexicographic reinterpretation work; furthermore, they concentrate often more on the presentational aspect of the entries (e.g. typographic conventions, etc.) than on a definition of the types of phenomena to be described. Although lexicographic description is to a certain extent a classificatory activity, it is by no means clear at first sight, from the end product (and even from many instruction manuals) which facts are underlying the classifications used in the dictionary.

Often lexicographers do not follow in all detail the instruction manual, or it is not sophisticated enough to cater for the situations a lexicographer may encounter in practical work. Syntactic consistency of article structures will be ensured, in future, in those dictionaries which have been produced with the support of a computational system for structural control. One such tool is the GestorLEX system which has been produced by a Danish software house.

[6] This is one of the reasons why we have not opted for a solution which would attempt to make use of one general text structure model for all articles of a dictionary. Such model would necessarily be rather general (and then most likely not cater for all indications potentially found in the dictionary) or it would be very complex and then difficult to handle in a concrete parsing exercise.

term is *Angabe*. Such individual elements may have particular relationships and interdependencies with other elements. In particular, scope (or: validity) relations play a role in this context: we have to determine whether a statement about a linguistic property of a lexeme is valid throughout the article or overridden by a piece of information given later in the same article. This is most relevant in dictionaries with a complex microstructure, such as those which use *Nestbildung* in German, or *regroupement* in French lexicography or other devices for the integration of subentries into the main entries. Insufficient knowledge about the scope of individual (types of) items can lead to interpretation problems in hierarchically or recursively organized articles: the interpretation of article structures in dictionary parsing must allow to reconstruct the scope of each statement within a given article.

The following are some examples from OALD: In the article s.v. *archives*, which has a subentry *archivist*, the meaning description "(*place for keeping*) *public or government records; other historical records*" refers to the main lemma, *archives*, reproduced in Fig. 2, whereas the meaning description "*person in charge of* ~" refers to the sublemma *archivist*.

ar·chives /'ɑ:kaɪvz/ *n pl* (place for keeping) public or government records; other historical records.
archi·vist /'ɑ:kɪvɪst/ *n* person in charge of ~.

Duce /'du:tʃeɪ/ *n* (I) leader (esp as used of Mussolini /ˌmʊsə'li:nɪ/ (1883–1945) Italian Fascist leader).

Fig. 2
Articles *archives* and *Duce*

The morphosyntactic indication "*n pl*" (which describes the main lemma as being a *plurale tantum*) has to be overridden by the morphosyntactic code "*n*" s.v. *archivist*, since with this word singular and plural are possible. The ~ takes up the form of the lemma *archives* in the meaning description of the embedded derivative *archivist*, but otherwise the sublemma does not "inherit" any particular information from its main lemma.

A more specific (and anomalous) case are glossations within articles: in this case, the explanatory item (e.g. a morphosyntactic or a phonetic item) does not refer at all to a unit with lemma status: in the article s.v. *Duce*, given in Fig. 2, the pronounciation item given within the semantic comment does not refer to the main lemma (*Duce*), but to an immediately preceding word of the semantic comment itself, the name *Mussolini*.

2.1.2. The method: path analysis

We distinguish two types of textual (and in part typographic, non-textual) items in dictionary text: one type conveys the actual linguistic description (e.g. a meaning description), the other is an indicator of the structure of an article. The latter type is more often non-textual (e.g. delimiter signs) and primarily allows to determine the scope of the items within a given article. Items which convey linguistic information can easily be represented, in the output of the transformation or extraction, in the form of attribute-value pairs as used in unification-based formalisms.

We describe both sorts of items in terms of their *item type* (function of an item, e.g. the type "meaning description") and of their *item value* (the actual textual form, e.g. the text of a meaning description).

Our analysis produces the following types of results:

- complex articles can be reduced, for further detailed analysis, to a few structurally relevant elements;
- an explicit description of textual interdependencies is given; this allows, for translation into an attribute-value format, to "collect" the relevant items for main lemmas and sublemmas, keeping track of inheritance and overriding in this process of collection of material;
- the recurrency of types of article structures can be checked throughout the dictionary and thereby the amount of "well-formed" and "ill-formed" articles can be identified;
- the results of the analysis serve as the basis of the specification of extraction and transformation rules.

Without going into all details, we now schematically describe the analysis of an article; taking as an example the article s.v. *archer* from the OALD.[7]

archer /'ɑ:tʃə(r)/ *n* person who shoots with a bow and arrows. **arch·ery** /'ɑ:tʃərɪ/ *n* [U] (art of) shooting with a bow and arrows.

Fig. 3
Article *archer*

[7] For a detailed description of the method, see Heyn (1992).

```
(ent :h "archer" (hwd "archer")(pr (ph "\"A:tS@(r)"))(hps :ps "n"
  (hsn (def "person who shoots with a bow and arrows")
       (cd (cp "arch|ery"(pr (ph "\"A.tS@rI"))
          (cps :ps "n" :cu "U"
             (csn (def "(art of) shooting with a bow and
                        arrows")))))))
```

For practical reasons we have reformatted the SGML-like articles of the electronic edition into lists of LISP expressions. Figure 3 reproduces the article as we find it in the printed version of OALD; the illustrations in Fig. 4 contain the representation of the article in a LISP expression, as well as a tree-like schema of its hierarchical structure.

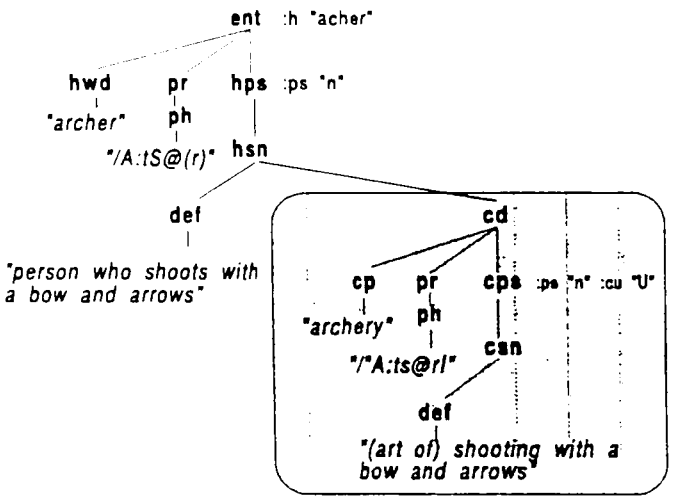


Fig. 4
LISP coded version of the article *archer* and its hierarchical structure

This latter figure shows clearly the recursive nature of the article: the part of the article which is devoted to the derivative (*archery*, in the grey box in Fig. 4) has almost the same internal structure as the main article. Along with this structure isomorphism, we find out about the functional equivalence of the nodes ent (entry) and hwd (headword) and cp (compound/desivetive), hps (headword part-of-speech) and cps (compound part-of-speech). This parallelism supports the transformation of the complex article, when reusing it for NLP purposes, into *two separate* simple articles.[8]

[8] This *flattening* has been carried out regularly in our transformations. Of course an

Furthermore, the illustration shows clearly the two types of nodes representing the two types of items introduced at the beginning of this paragraph, namely *linguistic information indicator items* and *structure indicator items*:

- *Information indicators*: items which convey linguistic information come as nodes and their terminals are linguistic "values"; cf. `hwd`, `cp`, `ph` or `def`. Alternatively, they have attribute-value pairs attached, such as in the case of `hps` or `cps`.
- *Structure indicators*: items indicating the textual structure of the article come as non-terminals with one or more daughters, but mostly without linguistic attributes, such as `cd`, `hsn` or `csn`.

We have analyzed in detail the structure and hierarchical organization of the the OALD articles in order to arrive at rules for transformation of the articles into an attribute-value pair based representation. For example, one rule says that a partial article structure graph containing the sequence of nodes `ent - hwd` {*terminal element*} is transformed into a partial attribute-value-structure of the form [`form`: {*terminal element*}].[9] Figure 5 shows the output of a complete path analysis of the article s.v. *archer*, i.e. of the application of several rules of the above type. It also shows the partial attribute-value-structures which are combined in a top-down manner along the relevant path, leading to the full set of linguistic descriptions at the bottom.

2.1.3. The tools: computer-assisted analysis of dictionary text structure

As a first step, the formulation of rules for the description of textual structures of dictionary articles has to rely on *manual* metalexicographic work. We have not yet implemented fully automatic transformation rules on the basis of path analysis, which could however now be done, once a complete set of structural descriptions is available.

explicit trace has to be kept of the fact that there has been a derivative entry embedded into another entry. The embedding is the presentational means used by the lexicographer to indicate that there is a derivational relation (affecting morphology, syntax and semantics of *archery*) between the derivative and the base. The same *flattening* of recursive dictionary structures is done in the ACQUILEX reformatting procedures, cf. Calzolari *et al.* (1990). In the CLOS representation we have generated, the nature of the embedding (compound/derivative vs. idiom group) in conserved and individual entries have pointers to each other ("my-derivatives:" vs. "derived-from:").

[9] For attribute names, we basically use those proposed within the ACQUILEX-Project (Calzolari *et al.* 1990); ACQUILEX has as well analyzed the OALD; however, the ACQUILEX analysis of this dictionary is not as detailed as ours, since the goals of the two projects are different. Furthermore the ACQUILEX report in question aims at establishing one general "template" for the structure of most of the articles of the OALD.

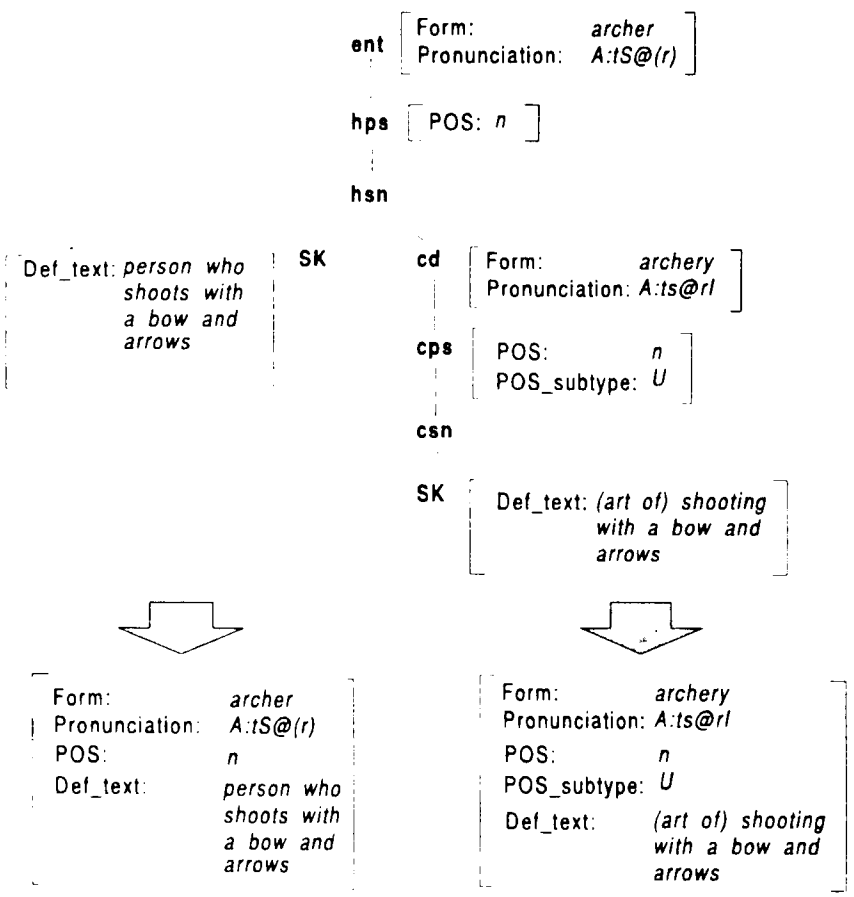


Fig. 5
Path information

Hypotheses about types of structure graphs and rules for automatic parsing of articles and for the collection of information along the relevant "paths" have to be checked against large quantities of material from all over the dictionary. This process (a sort of "explorative partial parsing" across the dictionary) can only be carried out with computational tools. On the basis of the hypotheses derived from our general knowledge about article structure and from a number of hand-analysed sample articles, string handling programs are implemented in `awk`, `sed` and similar programming languages and applied to all

articles of the dictionary. The results of the application of these programs allow for further refinement and modification. The programs are mostly parametric: types of structures, denoted by a list-like expression of embedded structure and information indicator items, are specified, and the tool extracts from the whole dictionary all those structures which follow the model specified. One of the advantages of the use of string handling languages is that they support a prototyping-like procedure. Programs can easily be refined until they provide the expected results. This explorative tool development is much like "filtering" of the dictionary according to changing and subsequently refined criteria. Simple counting of the entries following a certain type of article structure model allows to assess the coverage of the procedures.

By filtering articles which have a particular structure, it is possible to get statistical data about the repartition of the different types of article structures over the dictionary. Knowing about different structural types, we can further modularize the extraction rules (depending on structure types) and make them more specialized and more selective. In the case of the OALD, around 10% of the articles of the electronic edition deviate from any regular structure type which we could determine for this dictionary; this means that one out of ten articles of OALD 3e would cause trouble in extraction or transformation: for example, nodes are missing or do not have any function, etc.

2.2. Linguistic description and descriptives used

We have so far analyzed article structure. Reasons for this include the necessity to know about the access path leading to a certain type of item in a dictionary article, as well as the fact that the structuring of an article is in itself a presentational means used by the lexicographer to express certain facts about lexical items, such as e.g. their derivational relatedness.

Our second subtask is to find out about the types of linguistic information indicator items in the dictionary. They are, at the level of microstructure, the devices used by the dictionary authors to present their linguistic analyses.

In view of reusing the descriptive results presented by the dictionary authors in the dictionary, we need to do a "post-hoc" analysis of the relationship between the use made of the devices in question and the types of linguistic phenomena which the devices are used to describe. Ideally, we would expect that every type of linguistic phenomena which the authors of the dictionary found worth recording, would be described by exactly one descriptive device and its presentational (e.g. typographic, etc.) counterpart. We would furthermore expect that this very device was only used to describe this one fact, and that all types of facts are univocally described in this way. This situation

would allow us to search for one given type of device (e.g. in the form of a given textual markup or constellation of markups) to get out of the whole dictionary all and only the lexical items which have the property described by the device in question.

Unfortunately, this is not the case. A number of descriptive devices are polyfunctional, i.e. they are used to indicate several, unrelated phenomena; for some phenomena, more than one (synonymous) devices are used, other phenomena are not described explicitly but implied by the use of descriptive devices pointing to other phenomena. Similarly, the relationship between types of descriptive devices and actual instances of the use of these devices need to be explored: sometimes, the programme of the dictionary would foresee there to be an indication of a given type, but such indication lacks (by error) in some individual articles; or a given device seem to have been coined for the description of a specific instance of a class of phenomena, and other instances of the same class are treated by another, maybe more general device: often the frequency of use of a given device allows to assess its status: if a given device is used once or twice in the description of 52 000 items, it may be an ad-hoc creature of a lexicographer.

The following paragraphs are devoted to a more detailed analysis of the relationship between facts described and descriptive devices used.

2.2.1. Missing or implicit information

The analysis of a sufficient number of articles allows to infer the basic principles of the editorial programme of the dictionary in question, i.e. to know which types of information can be expected in which types of articles (e.g. depending on the category of the lemma).

However, in some articles the expected information is not given explicitly. This may have two reasons: the use of implicit devices or simply an error. A certain type of information may lack consistently and it may then be possible to infer it by rules; in this case, the information is implicit and we have to reconstruct the "rules" or "defaults" which the human user is supposed to apply when reading and mentally "processing" the dictionary article. For example, nominal derivatives with the ending *-tion* do not have a category mark in OALD; they are however marked for countability/uncountability. Since such countability marks only apply to nouns, we can reconstruct the category of the *-tion* nominalizations on this basis.

Quite frequently, information lacks unsystematically. Almost none of the item types is consistently present throughout all relevant articles of OALD. This fact has an impact on the interpretation of extraction and transformation

results. Here are some examples from OALD: 12% of all subentries of the dictionary (compounds, derivatives, idioms) do not have a pronunciation item, and 11% of all verbs do not have an item indicating their syntactic construction potential through a "verb pattern".

Computational tools have different relevance for the discovery and treatment of missing and implicit information: it is easy to detect structural "holes" in the analyzed dictionary articles, but it is near to impossible to mechanically detect implicitness.

The situation for the treatment of such cases is almost inverse: if there are clues allowing to formulate rules for explicitation of implicit descriptions, these are easy to apply automatically. Idiosyncratic gaps in the description are much harder to fill by means of rules. Often, a case-by-case treatment is needed; depending on the relevance and quantity of the material to be amended, the amount of missing information may have an influence on the overall assessment of the use of the respective transformation and extraction exercises.

2.2.2. Polyfunctional items and synonymous value names

We call *polyfunctional items* those where functionally different types of information indicator items have the same device. For example, OALD uses the same name to mark *pluralia tanta* and irregular plural forms. This is typically a problem of the use of presentational and especially typographical devices in the dictionary; it is not very easy to detect, since in many cases it requires manual cross-checking of the lists of partial article texts extracted from the dictionary. However, if the same names of item types occur in lists of entries extracted according to different criteria (i.e. on the basis of different structural environment types), this may indicate polyfunctionality of the respective item names.

A practical reason for the polyfunctionality of descriptive and presentational devices in dictionaries is the mismatch between what lexicographers have to say and what the language of typographic and non-textual devices allows them to say: a lexicographer would like to make more distinctions than he/she has means to express; recursive microstructures are particularly problematic in this respect, if e.g. one wants to copy the use of fonts from the main articles to subentries, but which says that the respective item is part of a subentry.

The inverse situation to polyfunctionality can also be found: *synonymous value names*. In such cases, two or more different names are used to describe occurrences of the same linguistic phenomenon. For example, OALD has *pred*

and *predadj* to denote predicative adjectives. Occurrences of such synonyms are discovered by a study of the types of value names appearing in the dictionary; once identified, it is trivial to find all occurrences of the respective phenomenon. But a merging presupposes an extensive study of the phenomena and almost an entire linguistic reclassification of the facts under consideration.

2.3. Statistical significance of items

Many *item values* are very infrequent or even hapaces. Most such cases do not concern particularly idiosyncratic phenomena which could not have been described within any of the "normal" classes of phenomena used anyway in the dictionary; they are, on the contrary, mostly ad hoc descriptions which have not been cross-checked with similar cases during dictionary production. For example, OALD has 78 different items indicating the word class. 98% of the occurrences of these are however instances of the usual part of speech types "noun", "verb", "adjective" and "adverb"; the remaining 74 category types make together 2% of the occurrences. This situation is shown in Fig. 6, which also illustrates that statistical irrelevance often correlates with synonymous value names: at some point, a redundant value name has been newly introduced.

selected part of speech values	*count*	*percentage*
n	20985	53.44%
adj	7001	17.83%
vt	4870	12.40%
vi	2788	7.10%
adv	2736	6.96%
adv_of_degree	5	0.01%
adv_of_place_and_direction	1	0.0025%
pron	65	0.16%
pers_pron	6	0.01%
emphat_pron	2	0.0050%
emph_pron	1	0.0025%
reflex_pron	1	0.0025%
interr_pron	3	0.0076%
interr_adv	2	0.0050%
interr_adj	1	0.0025%

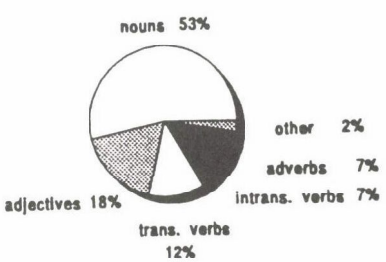


Fig. 6
Distribution of *part of speech* attributes

This type of problems can only be detected with realistic effort if computational tools are used. An inventory of all item values, for example of the word class indicator items, has been set up. All occurrences of the items have been collected and the frequency of the value names across the dictionary has been determined.

2.4. Summary: types of problems, computational support for detection and repair

Having discussed some examples of the most frequent problems encountered in the preparation of a transformation of the OALD into an attribute-value pair based format, we now try to summarize the types of problems encountered and to assess possibilites of discovering and repairing them by computational means.

The methodological and practical steps of the dictionary analysis start with the article structure. However, in the end, we are interested more in the reuse of the linguistic description conveyed mostly by information indicator items than in an account of the article structure itself. The structural analysis is thus a necessary precondition for the more linguistically oriented analysis of the reusability of the descriptive output.

The following typological summary of the most frequent errors encountered is also oriented towards linguistic information types and not towards article structure only. The following types of problems have been predominantly encountered:

1. *Obligatory items lacking* (cf. above, section **2.2.1**): once article structure patterns are determined, it is easy to find out the entries which lack certain items. However, it is almost impossible to automatically repair this problem: this would require new descriptive work. Consequently, a statistical assessment of the frequency of the problem and of the relation between complete and incomplete entries is often decisive for the inclusion of a certain type of item in the set of data supposed to be relevant for reuse.

2. *Implicit indications* (cf. above, section **2.2.1**): the reason for the implicitness of certain indications is the assumed "conviviality" of the dictionary user who can activate dictionary-external knowledge to "process" the entries of the dictionary. A particular type of implicitness is the use of default assumptions in the following sense: if no indication (e.g. of an irregular plural) is given, the applicability of a (sometimes as well implicit)

standard pattern (e.g. for plural formation) or of standard properties is assumed. These cases are very hard to detect and to repair by computational means.

3. *Polyfunctional items* (cf. above, section **2.2.2**): this problem comes down to a lack of descriptive differentiation among elements described by one and the same descriptive device. Tools for extraction of substructures and for comparing elements extracted allow to easily get access to the relevant material. However, reclassification is a descriptive task which needs to be carried out manually. Consequently, an assessment of the frequency of this phenomenon is useful before any transformation.

4. *Synonymous value names* (cf. above, section **2.2.2**): the fact that two value names are synonymous has to be established manually. But a list of all value names used anywhere in the dictionary along with the articles where these value names appear is easy to establish. Similarly, the comparison of the relevant articles can be prepared automatically, and once it is clear which value names can be "merged", this merging can be carried out by computational means as well.

5. *Frequency of descriptive devices* (cf. above, section **2.3**): it is of course easy to extract the relevant types of article structures and of items, and to run statistical tools on the results. Repair mechanisms have to rely on a (manual) reclassification of the data involved. And similarly, it is a matter of linguistic debate to decide whether or not to merge an infrequent type of description with another, more frequent one.

6. *Descriptive errors*: in the reinterpretation process, it may turn out that certain descriptive claims of the dictionary are not accepted by those who wish to reuse the dictionary. Such cases may require reclassification, however, detection and repair are both beyond the scope of computational tool support.

In the following illustration we summarize the above-mentioned types of problems and our experience concerning the usability of computational tools for detection and repair:

Type of problem (section)	Computational support for detection	Computational support for repair
obligatory items lacking (**2.2.1**)	yes, via structure types	no
implicit indications (**2.2.1**)	no	no
polyfunctional items (**2.2.2**)	(yes), initial support for manual work	(yes)
synonymous value names (**2.2.2**)	no	yes, after extensive preparation
frequency of descriptive devices (**2.3**)	yes	no
descriptive errors	no	no

3. Conclusion

This contribution deals with methods and tools for a detailed and exhaustive analysis of traditional dictionaries in view of their (re-)usability as information sources for NLP. We are thus interested in the amount and quality of linguistic information extractable from such resources. We discuss a method for the structural analysis of dictionary articles. This method, adapted from Wiegand's work in metalexicography, serves as a basis for two types of activities; the study of individual types of information in a given dictionary (realized through extraction of certain types of article substructures), and the formulation of transformation rules allowing to collect information along a given path through an article. Such a method is necessary for the reuse of those dictionaries of which the internal (article structure, textual deployment of articles) is not defined through a formal description, such as a DTD of SGML. This is the case with most dictionaries produced according to the traditional working methodology of lexicography.

We combine the structural analysis with a coherence check on structure types and with the statistical analysis of occurrences of certain types of items. The possibility of collecting evidence for the descriptive devices from all over the dictionary greatly facilitates this task and allows at all for quantifiable statements. The extraction and information routines are implemented as (in part parametric) pattern matching programs in string handling languages.

These tools, along with a descriptive linguistic assessment and an exact metalexicographic dictionary description allows to assess realistically the effort needed to re-represent a traditional dictionary for the purpose of NLP.

References

Alshawi, H. – Boguraev, B. – Carter, D. 1989. Placing the dictionary on-line. In: Boguraev – Briscoe.

Bläsi, Ch. – Koch, H.-D. 1993. Dictionary entry parsing using standard methods. In: This volume, 73–94.

Boguraev, B. – Briscoe, T. (eds). 1989. Computational Lexicography for Natural Language Processing. Longman, London – New York.

Calzolari, N. – Peters, C. – Roventini, A. 1990. Computational model of the dictionary entry. In: Preliminary Report. Projekt ACQUILEX. Esprit Basic Research Action No. 3030. Pisa, April 1990. ILC-ACQ-1-90.

Christ, O. 1990. Die Nutzbarmachung eines maschinenlesbaren Standardlexikons durch Transformation in eine CLOS-basierte lexikalische Datenbasis. Studienarbeit Nr. 924 am Institut für Informatik der Universität Stuttgart.

Hausmann, F.J. – Wiegand, H.E. 1989. Component parts and structures of general monolingual dictionaries: a survey. In: HSK 5.1: 328–60.

Heid, U. – McNaught, J. 1991. Eurotra-7 Study: Feasibility and Project Definition Study on the Reusability of lexical and terminological resources in computerized applications. Final Report. Under the responsibility of Ulrich Heid and John McNaught. Stuttgart.

Heyn, M. 1992. Wiederverwendung maschinenlesbarer Wörterbücher. Eine computergestützte metalexikographische Studie zur Wiederverwendung des Oxford Advanced Learner's Dictionary in NLP. (Lexicographica Series Maior 45). Niemeyer, Tübingen.

HSK 5.1 = Wörterbücher, Dictionaries, Dictionnaires. Ein internationales Handbuch zur Lexikographie. Eds: Hausmann, F.J. – Reichmann, O. – Wiegand, H.E. – Zgusta, L. Erster Teilband. Berlin – New York 1989 (Handbücher zur Sprach- und Kommunikationswissenschaft 5.1).

OALD 3 = Hornby, A.S. (ed.). 1974. Oxford Advanced Learner's Dictionary, 3rd edition. Oxford University Press.

OALD 3e = Hornby, A.S. (ed.). 1988. Oxford Advanced Learner's Dictionary, 3rd edition, Electronic Version. Oxford University Press.

Wiegand, H.E. 1989a. Arten von Mikrostrukturen im allgemeinen einsprachigen Wörterbuch. In: HSK 5.1: 462–501.

Wiegand, H.E. 1989b. Der Begriff der Mikrostruktur: Geschichte, Probleme, Perspektiven. In: HSK 5.1: 409–61.

Wiegand, H.E. 1990. Printed dictionaries and their parts as texts. An overview of more recent research as an introduction. In: Lexicographica 6: 1–126.

Wiegand, H.E. 1991. Über die Strukturen der Artikeltexte im Frühneuhochdeutschen Wörterbuch. Zugleich ein Versuch zur Weiterentwicklung einer Theorie lexikographischer Texte. In: Ulrich, G. – Reichmann, O. (eds): Historical Lexicography of the German Language. Volume 2: Lewiston – Queenston – Lampeter: Edwin Mellen 1991 (Studies in German Language and Literature Vol. 6 – Studies in Russian and German Nr. 3), 341–672.

Address of the author: Ulrich Heid, Matthias Heyn and Oliver Christ
Universität Stuttgart
Institut für maschinelle Sprachverarbeitung, Computerlinguistik
Azenbergstrasse 12
D-7000 Stuttgart 1
Germany

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 159-174 (1992-93)

TOWARDS A COMPUTERIZED HISTORICAL DICTIONARY OF DUTCH

JOHANNA G. KRUYT-JOHN J. VAN DER VOORT VAN DER KLEIJ

1. Introduction

The *Woordenboek der Nederlandsche Taal WNT* is a monolingual dictionary based on historical principles. It describes the Dutch vocabulary from 1500 up to the 20th century. The first fascicle was published in 1864. The dictionary will be completed in 1998. It will then comprise more than 100 000 columns of dictionary text (over 40 printed volumes). The *WNT* can be considered the Dutch counterpart of the *Oxford English Dictionary (OED)*, the *Deutsches Wörterbuch* started by the Grimm brothers, and the *Dictionary of the Swedish Academy (SAOB)*.

The *WNT* is currently being computerized. The *Electronic WNT* will be a text file encoded for information categories in SGML-format (Bryan 1988), similar to the electronic *New OED* (cf. Kazman 1986). That is, the running text will be interrupted by codes specifying the type of information a text fragment conveys.

A first prerequisite is a machine-readable version of the dictionary text. Since 1982, text processing facilities have been utilized in the production of the printed dictionary, which at the same time results in corrected machine-readable fascicles. The ca. 69 000 columns of dictionary text produced before 1982, however, are available in printed form only and are to be converted to machine-readable form. Until recently, a commercial firm was keying-in the text using word processors. Presently, optical character recognition OCR is applied. Correction of the output is computer-aided.

Lexicographical economy and structural ambiguity cause a corrected machine-readable version of the dictionary to be not suitable as the immediate input for the automatic encoding of information categories. An intermediate text revision process is required, which concerns an extensively corrected text file. A computer-aided procedure has been developed by which the revision can be performed with minimal risk of corrupting the corrected dictionary text, and by which the revisions can be checked efficiently.

Akadémiai Kiadó, Budapest

The automatic encoding of information categories is still in an experimental phase. A revised text file is input for a program that assigns start and end codes to text fragments indicating particular structural divisions within a dictionary entry.

The present paper describes the main components of the Electronic WNT project (section **3**). For reasons of clarity, first some characteristics of the printed *WNT* complicating the development of the *Electronic WNT* are discussed (section **2**). The paper concludes with an evaluation of the present methods (section 4).

2. The printed *WNT*: some features complicating conversion to electronic form

Compilation of the *WNT* started in the second half of the nineteenth century. The *WNT* therefore has not been designed to be converted to electronic form. The conversion is complicated by several characteristics of the dictionary. Here, we focus on some characteristics complicating the conversion to machine-readable form by OCR, and on some aspects of lexicographical economy preventing the machine-readable text file being suitable as the immediate input file for the automatic encoding of information categories.

Conversion of the printed text to machine-readable form by OCR is complicated by the uneven quality of the print from the old metal plates and by the complex graphical structure. Print quality is far from optimal due to different shapes of identical characters, broken characters, connected characters that should be separate, uneven brightness etc. Additionally, more than ten different print types have been used in the 504 fascicles printed before 1982. Graphical structure is characterized by numerous, frequently alternating type fonts, numerous special characters and symbols, and various indentation sizes which reflect the hierarchical structure within the entry (for more details, see Kruyt–van der Voort van der Kleij 1992). These factors cause the OCR-output file to be extensively corrected.

The term 'lexicographical economy' refers to all means a lexicographer applies for reasons of economy of space, like abbreviations, dashes replacing words, brackets reflecting various options for interpretation etc. Another aspect of lexicographical economy concerns quotation references. In the *WNT*, a quotation text is usually followed by the author, the title of the work, page number and a quotation date. In specific cases, however, the full reference is replaced by a partial one. For example, the quotation date, commonly enclosed by two square brackets, is left out when the title of the work includes

the date (which is the case for journals, almanacs etc.). Another example is shown by the following two quotations. The second reference "*Ald.*" ('there') means that the complete first reference, including the author "**HUBNER**", the title "*Koer.-tolk*", page "925 *a*" and date "[1732]", applies to the second quotation text as well.

> in 't keurvorstendom Saxen zyn twederly uytschotten ... die by uytschotsdagen beroepen worden, **HUBNER**, *Koer.-tolk* 925 *a* [1732]. In den ruymen uytschot bestaat de ridderschap uyt 60. personen ... en uyt 18. steden als Annaberg, Weissenfels enz., *Ald.*

The automatic encoding of information categories implies the automatic assessment of contents on the basis of graphical and lexicographical structure (cf. Kazman 1986). That is, form and structure are cues to contents. For this reason, it is particularly relevant to preserve the graphical characteristics of the original dictionary text, and to solve inconsistency caused by lexicographical economy. This implies, among other things, that dates are to be inserted when left out, as a date between square brackets is a textual feature that will be used as a marker for "end of quotation". A date with this particular form is therefore essential to the automatic separation of quotations and subsequently to the automatic encoding of the various components of the quotation reference (cf. Van der Voort van der Kleij–Kruyt 1992).

3. Towards a computerized *WNT*

3.1. State of the art

Main components in the Electronic WNT-project are (1) digitizing the volumes that are available in printed form only, (2) text revision in corrected dictionary text files, produced by either text processing or by OCR, and (3) automatic encoding of information categories. Focus is presently on activities (1) and (2), (3) still being in an experimental phase. An infrastructure has been developed by which the correction of OCR-output files, the text revision procedure as well as the check on text modifications, and the administration of these processes is computerized to a major extent.

3.2. From printed text to corrected text file

3.2.1. Text processing vs. OCR

Eight *WNT*-volumes have been keyed-in by a commercial firm using word processors. The output was manually corrected. In addition to the proper

text, some types of text revisions, supporting the future automatic encoding of information categories, were keyed-in. The revisions had previously been noted by WNT-lexicographers onto enlarged copies of the dictionary pages, and visually checked. Except for some minor points, this method is essentially similar to the one applied in the *New-OED* project (cf. Simpson 1986; Kazman 1986; Berg *et al.* 1988).

Mainly for financial reasons, the application of OCR was considered as an alternative. Institutional personnel, rather than a commercial firm, was envisaged to be charged with the conversion of the printed text into a corrected machine-readable version. After extensive experiments, a Kurzweil K5200 OCR system has been decided on, in spite of the rather poor quality of the output, due to reasons like those mentioned in section **2**. In order to correct the OCR-output files efficiently, a preprocessing program was developed, and manual correction at the screen is computer-aided. See Kruyt–Van der Voort van der Kleij (1992) for a more detailed description of the system developed.

3.2.2. Scanning and OCR

A Kurzweil K5200 connected with a Commodore 386SX-16 personal computer is used for scanning and OCR. The first stage is the development of a training-set. Pages are scanned, and, during scanning, all characters that are not unambiguous to the system are presented at the screen. Only characters that have a correct and clearcut shape are confirmed to the system as being correct. In this way, the system specifications for each character are refined according to the specific characteristics of the text to be processed. If structurally relevant special symbols cannot correctly be processed by the Kurzweil, a simulation is trained. For example, the two vertical lines "||", indicating the beginning of a quotation block, is trained as "!)!)", and is automatically converted into the original symbol in the automatic preprocessing phase (**3.2.3**). This also applies to the dash "—", which is trained as "+)". The training-set is stored and can essentially be used for all pages with similar textual characteristics.

After the training phase, portions of 16 columns of dictionary text are scanned with use of a form feeder, subsequently recognized, and then converted to a suitable format, in our case an ASCII-file. The portion of 16 columns (per column ca. 3500 characters on the average), proved to be a manageable quantity for computer-aided manual correction (**3.2.4**). Training, scanning, character recognition, and conversion require 2 minutes per column on the average.

The output is a running text file interrupted by codes, as shown by the following examples:

>>>UI, <8,R>Samenst. enz. <12,R>34

<8,R>grootte uitgezocht, <6,R> REINDERS, <8,I>Landb <8,R>2,201 [1893]. Uienzaad wordt in ons land ingevoerd voor het

>>><8,I>+)>>>Uienzweefvlieg <8,R> (3), zie de aanh. !)!) Kleine narcisvheg of uienzweefvlieg <8,I> (Euvierus strigatus),

(WNT, Vol. 17–3, column 34).

The codes indicate the beginning of roman ("R") and italic ("I") font only, as well as the relative height of following characters ("12", "8", "6", and other values). Note that codes only mark font shifts and height alterations. Many features of the printed dictionary (other fonts, special characters and symbols) cannot be represented in the OCR-output file. Note, however, the successful simulations "!)!)" and "+)".

The quality of the OCR-output text file is improved by automatic preprocessing (**3.2.3**), in order to reduce and facilitate the computer-aided manual correction at the screen (**3.2.4**).

3.2.3. Automatic preprocessing

After the PC non-ASCII values are converted to VAX non-ASCII values, the OCR-output text file is copied from the PC to the central hostcomputer Digital VAX 4000/500. The text file is now characterized by the extension ".icr".

Every night, the presence of OCR text files is automatically checked in a batch procedure, on the basis of the filename extension. If present, OCR text files are converted to VAX/VMS format and subsequently processed by a preprocessing program written in VAX-BASIC, which improves the proper text by controlling as many as possible systematic features. Essentially two types of operations are involved. Some concern the correction of errors consistently made in the OCR process, such as "DI" being corrected into "Dl", and "(11)" into "(II)". Focus, however, is on the correction of structurally relevant strings, such as correcting "QCU—" into "— QCU", "QCU" being a graphical code indicating the beginning of italic font.

Other operations concern structural improvements, rather than the correction of consistent errors. They include the insertion of fonts not, or not properly represented in the OCR text file. For example, for each text file small capital font is calculated on the basis of the digits in the OCR-codings (like those shown in **3.2.2**) and specific contextual features. Bold face is determined on the basis of either pattern recognition or a unique combination of textual characteristics. Roman font indications are removed, being considered the default font. Another operation concerns font code completion. The graphical

codes for italic font, bold face and small capital font are completed by end codes. For example, "QCU" means "start italic font", "ZCU" representing "end italic font". Special symbols, such as the two vertical lines and the dash (cf. **3.2.2**), are reconstructed. Structural indentations are checked on correct location and visualized by five subsequent copyright symbols. Running headlines are not yet removed in order to support manual correction, but standardized so as to be able to remove them automatically in a later stage of the computerization process. Words broken up at the end of a line by a hyphen are automatically connected. The number of spaces is regulated. Although different types of operations are involved in the preprocessing process, most rules in the program are based on textual patterning.

For an impression of the effect of the automatic preprocessing, in particular of the structural improvements, the example lines shown in **3.2.2** are presented here after they have been processed by the program.

*#$ KOPREGEL *#$ UI, Samenst. enz. 34

grootte uitgezocht, QKKREINDERSZKK, QCULandb.ZCU 2, 201 [1893]. Uienzaad wordt in ons land ingevoerd voor het

©©©©© — QCUUienzweefvliegZCU (3), zie de aanh. || Kleine narcisvheg of uienzweefvlieg (QCUEuvierus strigatusZCU),

The resulting text file has more or less the layout of the printed dictionary. Depending on the quality of the OCR text file, the preprocessing introduces 2000–3000 modifications in a file of sixteen dictionary columns.

The automatically preprocessed file is characterized by the extension ".conv". This file is input for the computer-aided manual correction.

3.2.4. Computer-aided correction

Correction comprises three phases. The first one is focussed on correction of the entire text, the second one particularly on textual features that are of primary importance to the future automatic text encoding in terms of information categories. First and second correction are performed by use of written instructions. The third phase concerns a final check by the present authors. Standard for correction is the printed dictionary text. Errors in the original text are not corrected, unless the error concerns an element essential to the future automatic encoding of information categories.

Correction is performed at the screen by use of VAX-terminals. A text file is interactively assigned to a particular corrector, either for first or second

correction. Each time a file has been read from disk and appears at the terminal, some automatic operations improve structural consistency. This concerns, for example, the regulation of spaces in quotation dates. For ease of correction, text structure is clarified by addition of screen attributes (reverse video, highlighting etc.) to the text enclosed by font codes. During correction, four buffers are visualized as windows at the screen: a text buffer including the text to be corrected, a comment buffer for comments by the corrector, a message buffer which informs the corrector about proper system messages as well as messages concerning correctness of specific textual features, and a command prompt buffer. This is essentially similar to the system described in **3.3.3**, except for two major differences: screen attributes do not replace font codes as the latter may be subject to correction, and corrections can freely be made in the text buffer.

Correction facilities include a menu supporting the insertion or modification of font codes. Specially defined keys support frequent operations, such as inserting or revising structural indentations, and connecting words broken up at the end of a line in cases where the hyphen has not been detected in the OCR-process. In the second correction phase, predefined keys support sophisticated searches in order to check all aspects essential to the automatic encoding of information categories, and frequently occurring errors originating in the OCR-phase that could not be corrected by automatic preprocessing. All these means aim at an optimal check on correctness of textual features during the first and second correction process. When the text file is completely corrected, either in the first or second correction phase, and the final session is finished, formal aspects of font codes are automatically checked and interactively corrected, because of their relevance to following stages in the project.

The final check and correction is facilitated by a connection between the line numbers in the text buffer containing the corrected dictionary text file and those in the comment buffer containing the second corrector's comments. This way, scrolling and searching is reduced to a minimum (cf. **3.3.5**).

The procedures described here are mainly based on textual patterning and are realized by use of the extensible VAX-editor EVE in combination with the programming language Text Processing Utility (TPU). First, second and final correction phase of each file is characterized by the file extension (".conv_cor", ".conv_cor_cor" and ".conv_cor_def", respectively). First correction requires 32 minutes per column on the average. The second and final correction procedures have recently been developed and tested. Total correction duration is estimated at ca. 45 minutes per column on the average.

3.2.5. Administration

The administration of the process is computerized to a major extent. Administrative data are automatically recorded in a separate file (sequential fixed file), by a VAX-BASIC program using input-files produced by the editing program.

Before scanning, filenames are constructed by a program. An example of a filename is "17S30001", specifying the volume number (17-3) and the number of the first out of sixteen columns in a text file (0001). These filenames are included in the administration file. Extensions specifying the phase of a text file in the whole process of computerization are automatically added to the filenames.

During correction, the various phases are recorded in the batch by use of the standard time provisions of the computer system, including the date and time of the beginning and end of the first and second correction, respectively. Correction durations can easily be computed.

This way of administration not only ensures a correct overview of the files in their various phases, but also facilitates planning the project by use of rather objective durations of operations.

3.3. From corrected text file to revised text file

3.3.1. Introduction

As discussed in section **2**, a revision process is required which concerns a.o. insertion of dates left out for reasons of lexicographical economy. The dates have to be keyed-in into the corrected dictionary text file at the right locations, because automatic insertion is impossible in most cases. A computer-aided procedure has been developed by which the revision can be performed with minimal risk of corrupting the extensively corrected text files. Moreover, the inserted dates are efficiently checked, both during and after the revision process. Final correction is computer-aided as well. This procedure is described in this section (for more details see Van der Voort van der Kleij–Kruyt 1992).

We use a VAX 4000/500 (running the operating system VMS) as central computer, connected with terminals. Advantages of our central computer system include: automatic back up every night, easy distribution of files to the correctors, automatic administration of the revision process (not yet implemented for this process, but cf. section **3.2.5**). The software tools used are EVE (Extensible Vax Editor), VAX-BASIC and VAXTPU (Text Processing Utility), products of Digital Equipment Corporation.

3.3.2. Preparation of text files

Prior to the text revision process, the layout of the corrected dictionary text files is changed by a program. The lexicographer's text is separated from the quotation block. At the start of each editorial line, a copyright sign plus a space are inserted. This sign enables the recognition of editorial text, which is not to be modified. The quotation block starts at a new line and its beginning is marked by two vertical lines, like in the printed dictionary. Output of this program is a file with the extension ".term". This file is input for a dedicated editor which only accepts files with this particular extension, so as to ensure that the input has the right form.

3.3.3. Restricted editing in a corrected text file

We developed a special editor to protect text fragments that must remain unmodified. This is relevant because we are dealing with an extensively corrected dictionary text file. Using the extensible VAX-editor EVE in combination with TPU, we made a dedicated editor for restricted editing. For this purpose, EVE has been extended with special procedures, which are programs that can be run during the editing session. Some are activated by starting the editor, others by pressing keys during the editing session. This editor performs a provisional check on the formal correctness of the dates to be inserted.

When the editor is started, special keys are defined, buffers are made, and the text file is read from disk and appears at the screen. In the text file, type fonts are represented by graphical codes (cf. section **3.2.3**). The graphical codes for bold, small capital and italic font are replaced by video attributes at the screen, in order to clarify the text structure (for italic for example: reverse video). Four buffers become visible on the screen in their windows: a text buffer (18 lines + status line), a work buffer (two lines + status line), a message buffer (one line) and a command buffer (one line).

Text window

The text window contains the dictionary text. This buffer is made unmodifiable. Scrolling and searching are possible, however. When the corrector has positioned the cursor at a correct location for date insertion, he presses the specially defined 'date key'. The program bound to the 'date key' checks wether the location is an appropriate place for date insertion. If the location is admitted, the cursor is placed in the work window. If not, the message "Not an admitted place for date insertion" is sent to the corrector, and the cursor remains in the text buffer. The program under the 'date key' first checks the

symbol at the cursor position and its context to the left and to the right. This context has to meet program conditions on text patterns. These conditions prevent in nearly all cases that dates are inserted at a wrong location. They also prevent that existing dates are altered. What defines an admitted location? A full stop in the quotation block followed by a space. One of the most important conditions for the text preceding the full stop is, that it does not contain an existing date. In the quotation in section **2**, for example, the full stop preceded by the date "[1732]" would not be admitted as a place allowing insertion. The full stop after "*Ald*" is an admitted location.

Work window

Insertion of revisions is performed via the work window only. The corrector may here key-in the date. The special 'reset key' empties this buffer and places the cursor at its original position in the text window. Pressing the 'confirmation key' results in an automatic checking of some formal aspects of the date keyed-in, for example the square brackets ("[1732" would not be accepted). If the formal conditions are not met, the corrector receives a message in the message window, indicating the error (in the case of "[1732", the message "The square brackets are incorrect"). The cursor is still in the work window. If the date to be inserted is formally correct, the text buffer is made modifiable and the date is transferred into that buffer. After that, the text buffer becomes unmodifiable again. During the transfer, the inserted date is surrounded by number signs (e.g. "*Ald.* #[1732]#."). The work buffer is emptied. The corrector may revise the inserted dates by repeating the procedure. The number signs help the 'date key' program to recognize inserted dates. They are also used as inserted date markers in the computational check (see **3.3.4**).

The work window is used for comments made by the correctors as well. This facilitates an efficient processing afterwards (see **3.3.5**). For making comments, the corrector uses the 'comment key'. Comments are related to problematic date insertions or detected text irregularities.

Message window

Messages to the corrector are shown in the message window. As this window has one line, only the last message is visible. Most messages to the corrector concern system messages (e.g. "Attempt to change unmodifiable buffer") or error messages (e.g. "Not an admitted place for date insertion").

Command window

The corrector gives his commands to the system in the command prompt window after pressing the 'command key'. For example the command "Wildcard find" for special search options.

Before the exit from the editor, the graphical codes are reinserted automatically. Three output files are produced: (1) the original dictionary text file provided with inserted, marked dates (file extension ".term_hc"); (2) a file containing all messages and all comments of the corrector (file extension ".term_mod"); (3) a file containing data provided by the computer system about starting time, ending time and duration of the editing session (file extension ".term_tim").

There is a provisional check on formal aspects during the editing session. However, a check on lacking or incorrect dates is still required. For this purpose, an exhaustive checking program has been written to minimize the final human check.

3.3.4. Computational check on text revisions

The checking program is written in VAX-BASIC. Input for the program is the revised dictionary text file. In the quotation block, the program first checks locations where a date might still be lacking, by use of conditions on text patterns and formal features relating to the beginning and end of a quotation. Next, correctness of the inserted dates is checked on the basis of combined conditions on text patterns and contexts. Date checking concerns formal aspects as well as contents of the date.

The output of the program is a text file reporting the name of each checked file, a list including the correctly inserted dates, the incorrectly inserted dates and the lines where dates probably should have been inserted. We will illustrate this with some examples (lines with more than 80 positions are split up).

The first example shows a correct insertion. The line number ("111") is followed by the string "MAT" (indicating that a date is inserted), by a small portion of the context to the left of the inserted date, by the inserted date and by a small portion of the context to the right. The asterisk is indicating the place of insertion.

```
111: MAT U 1918, 1396 QCUaZCU #$$[1918]#. De uniform van de
welpen be                                  *
```

The list of correct insertions may include some inserted dates at incorrect but not protected locations. A corrector might have incorrectly inserted a date, for example, after the strings "*Ned*" or "*Jaerb*" in the following quotation.

> ... zoo zullen zy verbeuren dien Wyn, en daer toe duizend Guldens, mitsgaders hunne neringe, *Ned. Jaerb.* 1750, 570.

We have not found this type of error until now. However, if present, these errors can be detected very easily, by reading a selective part of the output file.

Fourteen types of potential errors are distinguished, each marked by a bracket plus a type number. This enables searching for special error types. The following example shows the error type "14", a portion of 80 characters of the context to the left, the line number "716", the indication "MAT", a small portion of the immediate context preceding the insertion, the inserted date, and a message about the error type ("DAT. FOUT?" i.e. "Date incorrect?"). In this case, the corrector typed the wrong number "1899", which should have been "1889" corresponding with the date in the title.

```
{14    voor de veiligheid van den alleenloopenden voetganger
(QCUop een kermis), Haagsc
716: MAT h Jaarb.ZCU 1889, 29 #$$[1899]#. DAT. FOUT?, <>
cijfer |1889|                  *
```

Next example shows the function of the context of 80 characters. The message "DAT. ONGELIJK AAN VORIGE" (i.e. "Date not identical with the last one") in the output file reports that the inserted date is incorrect because the program has determined that the inserted date "[1938]" is not identical with the preceding date "[Kempen, 1938]". We can verify this by reading the context of 80 characters to the left in the output file.

```
{10   QKKCORNZKK., QCUBijv.ZCU [Kempen, 1938].
      Het uitgewalmde stroo dient om daken t
309:  MAT e dekken, QCUAldZCU. #[1938]#. DAT. ONGELIJK AAN
      VORIGE [Kempen, 1938]      *
```

The most frequent error type is error type number one, a missed insertion, illustrated by the following example. The indication "DAT?" and the message

"NOG TE DATEREN?" means: "Is a date lacking?". The date "$$[1855]" should have been inserted after "204 QCUbZCU".

{1 Goudvernis, dat noch door 't licht, noch door de lucht verschiet, QCUVolksvlijtZ
114: DAT? CU 1855, 204 QCUbZCU. Vroeger kwam het verschieten in de was → NOG TE DATEREN?

The output of this extensive checking program strongly supports the final correction.

3.3.5. Final correction

The final, human checking deals with the output of the checking program and a file containing all corrector's comments. In this phase, we use another extended version of EVE in which three main buffers are employed: a text buffer including the revised dictionary text, the corrector's comments buffer and a buffer containing the output of the checking program. Two buffers appear simultaneously at the screen. The first always is the text buffer, and the second is either the comment buffer or the buffer with the checking report. The comments and the output of the checking program can be handled quite efficiently by a link between the line numbers in the three buffers. We position the cursor at a numbered comment line or at a numbered report line, and by pressing the 'link key', the cursor is placed in the text buffer on the corresponding line. This line is highlighted and the comment or report line as well. This way, scrolling and searching is reduced to a minimum. Then we evaluate the comment or report and may modify the dictionary text.

The output is a dictionary text file in which textual revisions preparing the automatic encoding of information categories have been performed. The new file has the extension ".term_hc_kor".

3.4. From revised text file to automatically encoded text file

The corrected and revised dictionary text file will be input to a program that assigns start and end codes to text fragments, which indicate what type of information a particular text fragment conveys. Van Grunsven (1986) has investigated which information categories can be distinguished in the *WNT*, and how they are structured within the entry. The nature of the *WNT*, however, does not allow to detect automatically all distinguished information categories with acceptable error rates. It is not yet clear which information categories can efficiently be detected on the basis of graphical and lexicographical structure.

The text revision process supports the automatic encoding of at least the main structural information categories.

Experiments with recently published entries resulted in a successful program for the automatic, SGML-like encoding of some important information categories. Three main divisions within the entry (headword group, total sense block, and derivations/compounds block), as well as thirteen potential sense levels within the total sense block have been discriminated and encoded, including nesting of the hierarchically ordered sense levels.

The following method has been applied. While an entire entry is being read, the beginning of each paragraph is scrutinized for the presence of a sense block. If a sense block has been detected, the particular sense level is determined by a string pattern analysis on the beginning of each paragraph. Next, the number of the paragraph and a number indicating the level in the sense hierarchy are stored in a separate, temporary 'sense level string'. This string is a symbolic representation of the sense block locations and their sense levels in the entry. Once the reading of an entry is completed, the string is used to compute where (and if there are several tags required, in which order) the end tags are to be inserted in the dictionary text file. The computational results are stored in the 'sense level string'. This string is processed and results in the insertion of all start and (nested) end tags in the dictionary file. Finally, the encoded file is written to a new output file. The result is a dictionary text file encoded for these particular information categories (cf. De Bruin *et al.* 1991). Encoding of 750 columns of dictionary text required 33 CPU-seconds.

4. Discussion

To our project, the methods presented in this paper have substantial advantages over the earlier method of word processing and correction performed by a commercial firm. Although OCR is still a controversial tool for computerizing large scale lexicographical resources (cf. Berg *et al.* 1988; Malmgren 1988), a considerable reduction of costs per column has been achieved by this method. Of course, the reduction is related to the charge of our particular firm, and we need to be constantly attentive to better solutions because of the huge amount of columns to be digitized.

As each improvement is of prime relevance for the feasibility of a large scale project like ours, one may ask whether the correction of OCR text files could be combined with simultaneous linguistic structuring as suggested by Malmgren (1988), with the correction of errors in the original dictionary text (cf. **3.2.4**), or with improving internal consistency. Our experience is that this

is feasible to a limited extent only. The instructions grow too complicated. Errors in the original text often require extensive looking up procedures. Improvement of consistency is preferably performed by consistent automatic procedures rather than by inconsistent human correctors. For these reasons, we have chosen to postpone this kind of improvements to a later stage in the project.

Another major result concerns the organisation of the project. The production of machine-readable dictionary text by both OCR and word processing (new fascicles) at the institute implied a reconsideration of the various stages in the computerization of the dictionary. The text revision process would preferably be applied to all machine-readable text, irrespective of the origin. In contrast with the former method (cf. **3.2.1**), this textual revision process is presently performed after the text has been made machine-readable (cf. **3.3.3**). The computerization of this activity resulted in a considerable saving of time: 2 minutes instead of 6 minutes per column. As this process concerns 100 000 columns, this implies a considerable reduction of man-years. The former visual checking is replaced by automatic checking procedures during and after the revision process, which ensures an improved quality of the product. The revision would preferably be performed automatically. This is possible to some extent only, as exact and complete definitions of textual patterns cannot (yet) be formulated.

The availability of many text files stored on disks of the institutional computer supports research into textual features, allowing lexicographical hypotheses to be tested and textual patterns unknown so far to be revealed. This research has been relevant to the development of the computer programs for the processes described above, and is furthermore particularly relevant to the automatic encoding of information categories. The system described above also allows correctors working at home, using telecommunication facilities, without any essential change of the procedures. In conclusion, the present methods have resulted in an improved efficiency, a reduction of man-years and costs, and in an improved quality of the product.

Of course, the product aimed at is similar to the electronic *New OED*. The present methods, however, are different. Just like the *New-OED* project has been instructive to our project, we hope that aspects of our approach may be useful to other projects dealing with complicated structured text.

References

Berg, D.L.–Gonnet, G.H.–Tompa, F.W. 1988. The New Oxford English Dictionary Project at the University of Waterloo. UW Centre for the New Oxford English Dictionary, OED-88-01.

Bryan, M. 1988. SGML, An Author's Guide to the Standard Generalized Markup Language. Addison-Wesley Publishing Company, London.

De Bruin, H.J.B.A.–Van der Voort van der Kleij, J.J.–Kruyt, J.G. 1991. Algoritmen voor een dictionary entry parser voor het elektronisch WNT. In: INL Working Papers 91–102.

Kazman, R. 1986. Structuring the Text of the Oxford English Dictionary through Finite State Transduction. Doctoral dissertation. University of Waterloo, Data Structuring Group CS-86-20.

Kruyt, J.G.–Van der Voort van der Kleij, J.J. 1992. Towards a computerized dictionary of Dutch: from printed dictionary to correct text file. In: Kiefer, F.–Kiss, G.–Pajzs, J. (eds): Papers in Computational Lexicography COMPLEX'92, 203–10. Research Institute for Linguistics, Hungarian Academy of Sciences, Budapest.

Malmgren, S.-G. 1988. The OSA project: Computerisation of the Dictionary of the Swedish Academy. In: Literary and Linguistic Computing 3: 166–8.

Simpson, J. 1986. Opening Address: The New OED Project. In: Information in Data. Proceedings of the First Conference of the UW Centre for the New Oxford English Dictionary, 1–6.

Van der Voort van der Kleij, J.J.–Kruyt, J.G. 1992. Restricted editing in a corrected dictionary text file. In: Kiefer, F.–Kiss, G.–Pajzs, J. (eds): Papers in Computational Lexicography COMPLEX'92, 343–9. Research Institute for Linguistics, Hungarian Academy of Sciences, Budapest.

Van Grunsven, H.J.J. 1986. Schematisch overzicht van *WNT*-gegevens, tevens profiel van de beoogde databank van het Woordenboek der Nederlandsche Taal. Gardata, Leersum.

Address of the authors: Johanna G. Kruyt and John J. van der Voort van der Kleij
Institute for Dutch Lexicology INL
P.O. Box 9515
2300 RA Leiden
Holland

Acta Linguistica Hungarica, Vol. 41 (1–4), pp. 175–189 (1992–93)

PHONETIC SYLLABLES IN FRENCH: COMBINATORICS, STRUCTURE AND FORMAL DEFINITIONS

ÉRIC LAPORTE

0. Introduction

Syllabication algorithms have been studied for the purpose of hyphenation by computer (e.g. Désarménien 1986; Mañas 1987), since in many European languages syllabication and hyphenation have much in common. In French and in Spanish, for example, most people would not make any difference between the questions *What are the syllables in this word?* and *Where can I hyphenate this word?* However, for no language the two operations are completely identical: e.g. the French word *abri* has clearly two syllables, but cannot in principle be hyphenated, since typographers forbid hyphenation when it leaves one of the letters alone at the end of a line (*Code typographique* 1981). This kind of rule is motivated by readability or esthetic considerations. In addition to that, syllabication is a phonetic matter whereas hyphenation is an orthographical one, and in a given language the spelling system is never completely consistent with the phonetic system. In English, syllabication and hyphenation are very different (e.g. Liang 1983).

Our matter here is a linguistic one, so we syllabify phonetic and phonemic strings, regardless of spelling and hyphenation. When we use intuitions, we will carefully identify and discard those related to spelling or hyphenation, and concentrate on those related to pronouncing.

1. Intuitive pronounceability

Clear intuitions about syllables are a valuable starting point. One of them is that the existence of syllables has something to do with pronounceability. For example, in syllabified speech, we alter the normal utterance by inserting pauses [#] (in the phonetic sense, i.e. interruptions of speech signal): this alteration is always possible, for any pronounceable utterance, and the resulting intermittent utterance is always pronounceable, e.g. since *rédhibitoirement* [#redibitwarmã#] is pronounceable, so are the isolated syl-

Akadémiai Kiadó, Budapest

lables [#re#di#bi#twar#mã#]. This is why the notion of syllable is so general. An even more general intuition is that a sequence of phonetic symbols is pronounceable if and only if it is made of syllables pronounceable in isolation and ordered in pronounceable combinations: e.g. the unattested sequence *[#pastjãd#] sounds French; connectedly, the syllables [#pas#] and [#tjãd#] are pronounceable separately as well as in sequence. These intuitions have interesting features: first, they are not restricted to linguists but widespread among most kinds of speakers; second, they are likely to be felt by the speakers of any language.

They lead to another question: what is pronounceability? A phonetic transcription is made of phonetic symbols, but not any sequence of phonetic symbols is a valid phonetic sequence in a given language, it can be unpronounceable or unconceivable in that language, e.g. *[gldbwvi], and the speakers of the language may have clear intuitions about it. Note that pronounceability is not completely defined by universal abilities of human vocal tractus, but is relative to a given language: French speakers utter very easily such sequences as [drwa] (*droit*) and [plɥi] (*pluie*), which puzzle many an Italian or English speaker, but generally encounter a difficulty if asked to repeat [ɛwri] in It. *Euridice*. The sequence [ɛw] followed by a consonant does not appear in French words, even in foreign borrowings: whenever this sequence occurs in the original language, it is adapted into another phonetic sequence.

However, these intuitions are too fuzzy to provide a reliable basis to a formal definition of phonetic syllables or of pronounceability. Moreover, even when they are clear, they can be false and it would be useful to check them by confronting them with other data. Much theoretical work has been dedicated to syllable recently, of course always in connection with intuitions, but generally not in connection with lexical phonetic data (with exceptions, e.g. Vennemann 1990). We consider syllabic structure not only as a theoretical problem but also as a lexicological one (cf. Aubergé *et al.* 1988) and we chose to confront systematically intuitions with lexical phonetic data. We made experiments on DELAP-FH[1] (Laporte 1988), a list of phonetic transcriptions of French inflected words, excluding proper names. Homonyms are transcribed only once and the total number of transcriptions is 218,700. The transcriptions were computed from the phonemic transcriptions of the dictionary DELAP,[2] through (i) automatic inflection and (ii) phonemics-to-phonetics conversion.

[1] An acronym for the LADL's phonetic dictionary of inflected forms with homonyms assembled together.

[2] An acronym for the LADL's phonemic dictionary.

The dictionary contains borrowings, so the experiments take into account exceptional words as well as others.

Phonetic strings are transcribed with the concern of maximal closeness to utterances, whereas phonemic strings are scholarly constructions. In phonemic strings, the notion of syllable is as relevant and useful as in phonetic strings, but even more difficult to define. Clear intuitions about syllables are attached to utterances, and so to phonetic transcriptions of them, but they get fuzzier when related to more abstract transcriptions. Moreover, conclusions about abstract syllables are expected to depend on the particular framework or background of the abstract transcriptions used. Finally, these theory and principles usually depend in turn on an abstract notion of syllable ... Phonetic representatives of utterances seem therefore to be the most adapted material for experiments.

In the sequel, phonetic sequences are written in the International Phonetic Alphabet (IPA); however, we apologize for writing uvular *r* as [r]. In the dictionary DELAP-FH, the transcription is narrow, especially if compared to that of publishers' dictionaries. In case of phonetic variants, the dictionary gives one of the variants, e.g. [kole] and not [kɔle] (*coller*), [fnɛtr] and not [fønɛtr] or [fœnɛtr] (*fenêtre*). In the latter case, the choice of the shortest variant produces many observable consonant clusters.

2. Characterization of pronounceable sequences

The notion of pronounceable sequence in a language relies on the intuitions of the speakers of the language, but a formal characterization or model would be useful. A list of the words of the language is not a satisfying characterization of the pronounceable sequences, no matter how exhaustive the list is. Phonetic sequences which are clearly not accepted as words of the language can be clearly pronounceable, e.g. *[#pastjãd#] for French.

In order to provide a characterization of pronounceable sequences in French, we will rely on the intuition that such a sequence is a succession of elementary pronounceable phonetic sequences ordered in pronounceable combinations. These elementary sequences may be syllables or other objects, but they may not be simply the phonetic symbols since consonants are not pronounceable in isolation.

For clarity we will adopt the mathematical terminology of word theory. A finite set of objects is called an *alphabet* when one considers sequences of them: two examples of alphabets are the set of phonetic symbols and the set of all possible syllables. A sequence of elements of an alphabet is called a *word*. The

set of all possible finite sequences of symbols of an alphabet A is noted A^*, and any subset L of A^* is called a *formal language* on A. With these terms, if A is the alphabet of all phonetic symbols, A^* is the set of all finite sequences of phonetic symbols in any combinations; the set of all phonetic transcriptions of all French words is a formal language on A, and can be characterized by the list of its elements; the set of pronounceable sequences is another formal language P on A, but no formal characterization of it is known, not even the list of its elements, which is too large.

A more realistic characterization of the set P of pronounceable sequences would take the following form:

- an alphabet S of "elementary" phonetic sequences, e.g. syllables; the internal structure of these "elementary" phonetic sequences is restricted by some constraints (internal constraints): e.g. some consonant clusters are possible in a language and not in another;
- a formal language T on S which describes how the elements of S can assemble together (combinatorial constraints). The elements of P itself are deduced from those of T by concatenating the elementary phonetic sequences.

For example, if the alphabet S is the set of syllables, the word *apprivoiser* is described by a word of T which is a sequence of 4 elements of S: `[a]`, `[pri]`, `[vwa]`, `[ze]`; the corresponding word of P is the phonetic transcription `[aprivwaze]` which is a sequence of 9 phonetic symbols in this case. There is an analogy with Chinese writing where each ideogram is generally pronounced as a syllable.

The most natural choice for the alphabet S is (i) the set of syllables, e.g. `[par/ti/ky/la/rism]` for *particularisme*, but it is not the only possible instance of the scheme above. We investigated three other reasonable choices for S:

- (ii) so-called antisyllables, e.g. `[#pa/arti/iky/yla/ari/ism#]`;
- (iii) vowel-consonant(s) sequences, e.g. `[#p/art/ik/yl/ar/ism#]`;
- (iv) consonant(s)-vowel sequences, e.g. `[#pa/rti/ky/la/ri/sm#]`.

In each of the four cases, combinatorial constraints tell how the elements of S can assemble together. These constraints are the simplest in the case of antisyllables. Medial antisyllables can be distributed into 169 classes $C_{l,\,r}$ according to their left l and right r vowels, e.g. `[arti]` is in $C_{\texttt{[a]},\ \texttt{[i]}}$. Initial antisyllables make up 13 classes B_r according to their right vowel, e.g. `[#pa]`

is in $B_{[a]}$. Final antisyllables make up 13 classes E_l according to their left vowel, e.g. [ism#] is in $E_{[i]}$. A sequence is in T if and only if:

- it begins with an element of B_r,
- it possibly continues with elements of $C_{l,\,r}$,
- it ends with an element of E_l,
- the element of B_l is followed by an element of $C_{l,\,r}$ or E_l,
- and every element of $C_{l,\,r}$ is followed by an element of $C_{r,\,s}$ or E_r.

The combinatorics of antisyllables is restricted by purely algebraic constraints which are of no linguistic interest. In contrast, the combinatorics of syllables, that of vowel-consonant(s) sequences (VC^*) and that of consonant(s)-vowel sequences (C^*V) are language-dependent linguistic data (cf. sections **4**, **5**, **6**). Little is known about these combinatorial constraints.

There is no clear-cut limit between internal constraints and combinatorial constraints in general: this categorization depends on the choice made to identify S and T. For example, in French, a nasal vowel is generally not followed by a semivowel ([j] [ɥ] [w]) in the same word: *[ãwa] is unattested. This constraint between a vowel and the following consonant is an internal one in the case of antisyllables (ii) and VC^* sequences (iii), but a combinatorial one in the case of syllables (i) and C^*V sequences (iv).

In the sequel, we do not consider phonetic sequences lapping over two words, like [lɔ̃ŋvy] (*longue-vue*). We will thus refer to word-initial sequences as initial, etc. The figures in the paper will obviously be higher when one considers sequences lapping over word limits and new phonetic variants will come up. For example, the consonant cluster [ŋv] attested in [lɔ̃ŋvy] is unattested inside French words.

We extracted from the dictionary the set S in the 4 cases above. The case of syllables (i) is the most difficult, since we have no formal definition of the syllable and no received syllabication algorithm. The set of syllables is thus described in section **6**. The following table (Fig. 1) shows the size of S in the case of antisyllables (ii), VC^* sequences (iii) and C^*V sequences (iv). The line for isolated sequences in the table stands for the case when an entire word happens to be an element of S: these words are interjections (*brr! pff! pfft! pst!*) and elided words, which are not in DELAP-FH.

	(ii) Antisyllables	(iii) Vowel-consonant(s)	(iv) Consonant(s)-vowel
Initial	1178	216	1183
Medial	9189	2159	2483
Final	589	593	138
Isolated	5	5	5
All types	10961	2438	2794

Fig. 1
Size of the alphabet S in cases (ii), (iii) and (iv)

Internal and combinatorial constraints are not completely arbitrary or irregular, but partially consistent with acoustic and articulatory categories of phonetic segments. For example, in a consonant cluster, only the first and the last consonants can be a semivowel:[3] [ʒoajri] *joaillerie*, [bwa] *bois*, but *[abwvi] is unpronounceable. The three phones [j] [ɥ] [w], which are acoustically and articulatorily related, behave in the same way as far as this rule about consonant clusters is concerned. In French, this partial consistency is more limited than one can expect. The history of the language brought about distributional gaps: e.g. all full consonants,[4] except [g], can combine with [j] into a pronounceable and attested initial consonant cluster, but the sequence [gj] is not attested as initial cluster, though it is pronounceable in association with most vowels. In addition, foreign borrowings introduced rare sequences which distort the symmetry of the system. Some of these borrowings are marginal and could be neglected for the sake of symmetry, but many others are ancient, frequent and not obviously recognizable as borrowings. For example, take the following rule: when a consonant cluster begins with [l] or [r] followed by at least two other full consonants, then it contains only two other full consonants, and the last one is [l] or [r], like in [filtr] *filtre*. We extracted from the dictionary the exceptions to this rule. All are borrowings or learned words, but some of them are frequent and easy to pronounce: *absorption*, *marxiste*, *perspicace*, *solstice*, *superstitieux* ... There are 111 exceptions (each verb counts as one; the list is given in below).

[3] We include semivowels in the list of consonants; we consider glides as semivowels, even between a vowel and a consonant.

[4] A full consonant is a consonant which is not a semivowel.

absorption, absorptions
absorptivité, absorptivités
adsorption, adsorptions
antarctique, antarctiques
arctique, arctiques
austromarxisme, austromarxismes
calcschiste, calcschistes
circumantarctique, circumantarctiques
coarctation, coarctations
démarxiser
désorption, désorptions
dolce
dolcissimo
feldspath, feldspaths
feldspathique, feldspathiques
feldspathoïde, feldspathoïdes
feldwebel, feldwebels
hallstattien, hallstattiens
hallstattienne, hallstattiennes
hertz
hertzien, hertziens
hertzienne, hertziennes
holantarctique, holantarctiques
holarctique, holarctiques
hornblende, hornblendes
horst, horsts
hypermnésie, hypermnésies
hypermnésique, hypermnésiques
hypersplénisme, hypersplénismes
hyperstatique, hyperstatiques
hypersthénie, hypersthénies
hypersthénique, hypersthéniques
infarctus
interpsychologie, interpsychologies
interstellaire, interstellaires
interstice, interstices
interstitiel, interstitielle, interstitielles, interstitiels
kammerspiel
karst, karsts
karstique, karstiques
kilohertz
mégahertz
maëlstrom, maëlstroms
maëlström, maëlströms
malabsorption, malabsorptions
malstrom, malstroms
marxien, marxiens
marxienne, marxiennes
marxiser
marxisme, marxismes
marxiste, marxistes
marxologue, marxologues
millivoltmètre, millivoltmètres
oersted, oersteds
oerstite, oerstites
paramètre, paramètres
perchman
perchmen
percnoptère, percnoptères
perscruter
perspectif, perspectifs
perspective, perspectives
perspectivisme, perspectivismes
perspicace, perspicaces
perspicacement
perspicacité, perspicacités
perspiration, perspirations
physisorption, physisorptions
quartz
quartzeuse, quartzeuses
quartzeux
quartzifère, quartzifères
quartzifier
quartzique, quartziques
quartzite, quartzites
quartzitique, quartzitiques
réabsorption, réabsorptions

résorption, résorptions
recordman, recordmans
recordmen
rinforzando
scherzando
scherzo, scherzos
sforzando
solstice, solstices
solsticial, solsticiale, solsticiales
solsticiaux
sportsman
sportsmen
sportswoman
sportswomen
subantarctique, subantarctiques
subarctique, subarctiques
superstar, superstars
superstitieuse, superstitieuses
superstitieusement
superstitieux
superstition, superstitions
superstrat, superstrats
superstructure, superstructures
surstabilisation, surstabilisations
surstimulation, surstimulations
surstockage, surstockages
terzetti
terzetto, terzettos
tolstoïser
tolstoïsme, tolstoïsmes
turkmène, turkmènes
ulster, ulsters
verste, verstes
voltmètre, voltmètres

3. Antisyllables

Medial antisyllables stretch from a vowel to the next vowel. Initial antisyllables stretch from a word boundary to the first vowel, and final antisyllables from the last vowel to the next word boundary. Consequently, the three sets are pairwise disjoint.

Antisyllable boundaries are located in the center of vowels, i.e. in the most stationary regions of speech. This means that provided the sequence of antisyllables is consistent with the formal constraint of section **2**, the acoustic contents of an antisyllable are relatively independent of the contents of adjacent ones. This property might make antisyllables an interesting unit for speech synthesis and recognition. Unfortunately, they are much more numerous than VC^* sequences and C^*V sequences, and even than syllables, at least when one excludes syllables lapping over word limits.

4. Vowel-consonant(s) sequences

To obtain the list of VC^* sequences, we divided each word before each vowel, e.g. `[#p/art/ik/yl/ar/ism#]`. For each word, the first sequence obtained is labelled as initial VC^* sequence but contains in fact the initial consonant cluster, here `[#p]`, since the first division occurs before the first vowel. (We

speak of a consonant cluster even in the case of an isolated consonant or of no consonant at all: empty cluster). The other sequences contain a vowel and a consonant cluster. This explains the small number of initial *VC** sequences (216) in comparison with that of final *VC** sequences (593).

We used the list of medial *VC** sequences to study the distributional constraints between a medial consonant cluster and the preceding vowel and to implement a model of them. If the 529 attested medial consonant clusters combined freely with the 13 French vowels, there would be 6877 medial *VC** sequences, but only 2159 (31%) appear in the dictionary. Some interdictions are systematic, e.g. a nasal vowel is never followed by a semivowel; other seem contingent or anecdotal, e.g. *[irʒw] is not attested, though [irʒ] (*virginal*) and [urʒw] (*bourgeois*) are. Note that [irʒw] might be attested in a proper name. We modelled the most systematic constraints in the form of a finite automaton which recognizes 6357 *VC** sequences out of the theoretical 6877 (92%). This automaton is shown in Fig. 2.

5. Consonant(s)-vowel sequences

The definition of *C*V* sequences is symmetrical with that of *VC** sequences, e.g. [#pa/rti/ky/la/ri/sm#]. For each word, the last sequence obtained after division is labelled as final *C*V* sequence but contains in fact the final consonant cluster, here [sm#], since the last division occurs after the last vowel. The other sequences contain a consonant cluster and a vowel. This explains the small number of final *C*V* sequences (138) in comparison with that of initial *C*V* sequences (1183).

We used the list of medial *C*V* sequences to make a model of the distributional constraints between a medial consonant cluster and the following vowel. If the 529 attested medial consonant clusters combined freely with the 13 French vowels, there would be 6877 medial *C*V* sequences, but only 2483 (36%) appear in the dictionary. Most interdictions seem little systematic, e.g. medial [ksu] is not attested except in proper names, though [ksy] (*sexuel*) and [kswa] (*aixois*) are. However, two interdictions are systematic: the semivowel [ɥ] is never followed by the vowel [y], and [j] is never followed by [i], except in [ji] (*failli*) and [nji] (*rejoignit*); this can be implemented in a simple finite automaton which recognizes 6722 *C*V* sequences out of the theoretical 6877 (98%). If we compare the percentages for *C*V* and *VC** sequences, we observe that *VC** sequences are more constrained than *C*V* sequences, and that the systematic constraints are stronger.

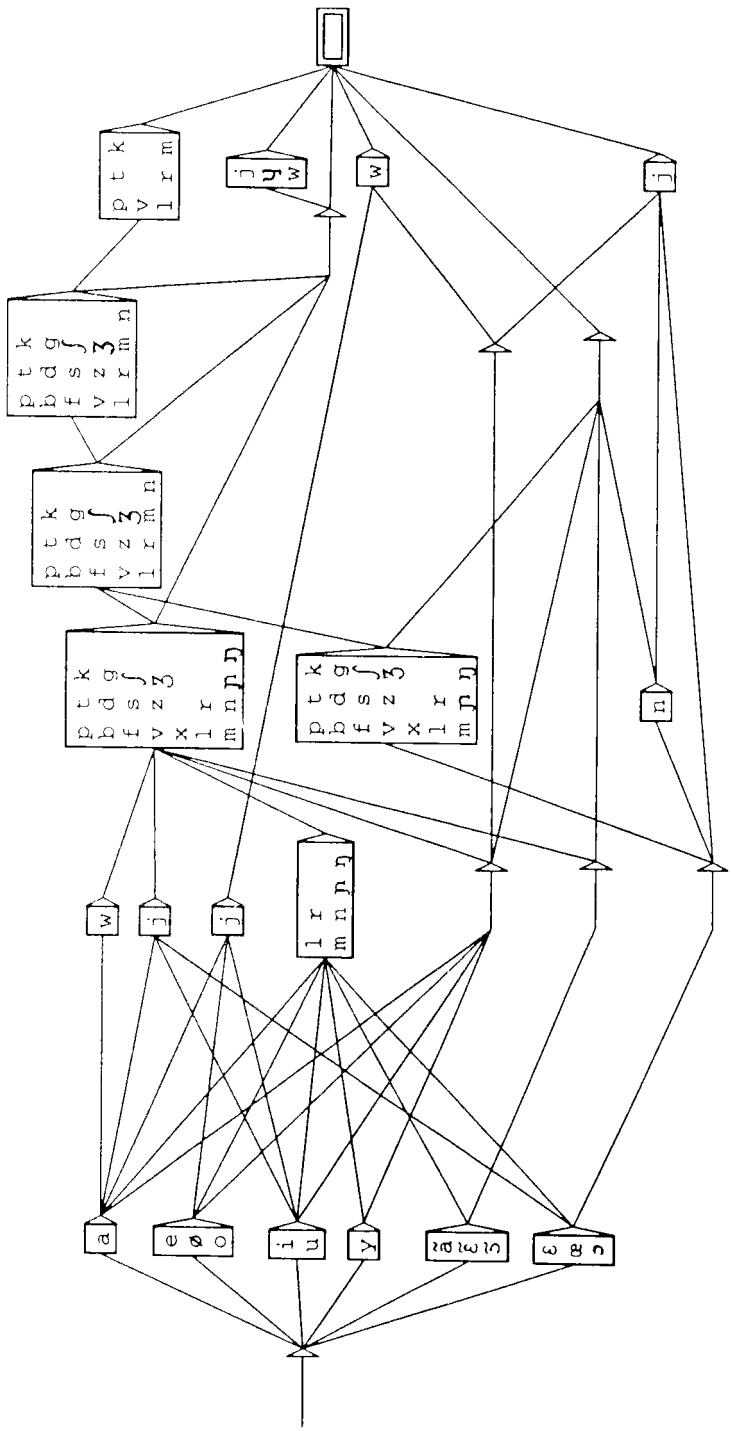

p t k
v l r m
j ɥ w
w
j
p t k
b d g f s ʃ v z ʒ l r m n
p t k
b d g f s ʃ v z ʒ l r m n
p t k
b d g f s ʃ v z ʒ x l r m n ɲ ŋ
p t k
b d g f s ʃ v z ʒ x l r m ɲ ŋ
n
w
j
j
l r m n ɲ ŋ
a
e ø o
i u
y
ɶ œ ɔ
ɛ ɐ ɔ

Fig. 2

6. Syllables

The main problem in syllabication is to locate the syllable boundary when a medial consonant cluster contains several consonants, e.g. to choose between [a/tlas] and [at/las] (*atlas*). In case of an isolated consonant or of a hiatus, syllabication is not controversial in French: [ra/pid] (*rapide*), [na/if] (*naïf*).

In order to discover criteria for locating syllable boundaries, let us go back to intuitions. Recall that apparently every pronounceable utterance can be syllabified into pronounceable syllables by inserting pauses, e.g. since *rédhibitoirement* [#redibitwarmã#] is pronounceable, so are the isolated syllables [#re#di#bi#twar#mã#]. But a pause may not always be inserted at any point of an utterance: e.g. if we divide [ʒo/a/jri] (*joaillerie*), one of the pieces is unpronounceable in isolation (*[jri]).

Now the pause inserted in an utterance always divides a medial consonant cluster in two (possibly empty) consonant clusters. Let us translate the assertion above in terms of consonant clusters. Apparently every VC^*V sequence in a pronounceable sequence can be divided into:

- a VC^* sequence which is the end of a pronounceable sequence,
- and a C^*V sequence which is the beginning of a pronounceable sequence.

For example, [ajri], which is a part of the pronounceable sequence [ʒoajri], can be divided into [aj#] and [#ri].

This is consistent with intuition and means that syllabication is possible. This suggests a test to check whether a given syllabication algorithm is consistent with intuition. If we could check whether a VC^* sequence is the end of a pronounceable sequence, and whether a C^*V sequence is the beginning of a pronounceable sequence, we could thus test a necessary property of syllabication.

Unfortunately, pronounceability is not a formal notion. But we can test an approximation of it, if we use a (large enough) dictionary. Assume that for any VC^* sequence which is the end of a pronounceable sequence, the final cluster appears as final cluster in at least one word, and assume the symmetric property for C^*V sequences. Then a good syllabication algorithm should divide any attested medial cluster into an attested final cluster and an attested initial cluster.

Both of the assumptions above are false: e.g. [ɛlts] is pronounceable in French in [fɛlts/pat] (*feldspath*), but [lts] appears at the end of no word; more spectacular, [gje] is perfectly pronounceable in [di/vyl/gje]

(*divulguiez*), but no French word begins with [gj]. We implemented the test anyway and tested 3 syllabication algorithms.

6.1. The first algorithm has little to recommend it:

> if a medial consonant cluster ends with a semivowel, divide before the semivowel, else divide after the cluster, e.g. [filtr/e] (*filtrer*), [parl/je] (*parliez*).

Once a medial cluster is divided, the right piece is always attested as initial cluster, but in 269 out of the 529 attested medial clusters (51%), the left piece is not attested as final cluster, e.g. [lr] from [kulr/a] (*coulera*).

6.2. The second algorithm is the one used in our phonemics-to-phonetics conversion algorithm (Laporte 1990) to determine the aperture of mid vowels:

> if a medial consonant cluster begins with a semivowel followed by another consonant, divide after the semivowel; if it begins with another sonant [l r m n ŋ] followed by a full consonant,[5] divide after the sonant; else divide before the cluster, e.g. [ʒo/aj/ri] (*joaillerie*), [par/le] (*parler*), [a/si/ste] (*assister*).

Here the left piece is always attested as final cluster, but in 156 out of the 529 attested medial clusters (29%), the right piece is not attested as initial cluster, e.g. [psjɔ̃] from [o/psjɔ̃] (*option*).

6.3. The third algorithm was designed to stand the test. For every medial cluster we extracted from the dictionary all the ways of writing it as the concatenation of an attested final cluster and an attested initial cluster. These data suggested an algorithm which produces an acceptable division in most cases:

> if a medial consonant cluster contains one of the symbols [p t k b d g f v] followed by one of the symbols [l r], say that these two symbols count as one; divide the cluster before the last symbol which is not a semivowel, e.g. [ku/ple] (*coupler*), [par/le] (*parler*), [op/te] (*opter*), [de/plwa] (*déploie*), [par/lje] (*parliez*), [op/sjɔ̃] (*option*), [eks/kly] (*exclu*).

This algorithm is similar to most syllabication algorithms based on tradition (e.g. Mañas 1987). Only 22 out of the 529 medial clusters (4%) are not divided into an attested final cluster and an attested initial cluster. Among these 22

[5] A full consonant is a consonant which is not a semivowel.

clusters, 4 contain the sequence [gj] which is not attested as initial cluster though it is pronounceable in association with most vowels. The other 18 clusters appear in 32 words only (each verb counts as one; the list is given in Fig. 3). These words are those for which the syllabication algorithm and the model of pronounceability are incompatible. The words are displayed so as to dedicate one line to each distinct pronounciation. When examining these words, one can notice that:

- either the phonetic transcription used in the processing is doubtful (e.g. [bejwar] for *baignoire*: the exact phonetic value of the cluster is unclear);
- or the pronounceability is questionable because of a rare cluster (e.g. [zɛmstvo] for *zemstvo*);
- or it is questionable whether the word is actually a simple word (e.g. *postscolaire*);
- or the model of pronounceability for initial clusters is questionable (e.g. for *jésuite*: it excludes [zɥ] because this cluster is not attested in French as initial cluster except in proper names).

singspiel, singspiels
baignoire, baignoires
égratignoir, égratignoirs
éteignoir, éteignoirs
peignoir, peignoirs
rognoir, rognoirs
saignoir, saignoirs
beefsteak, beefsteaks
reichsmark, reichsmarks
reichsmark, reichsmarks
reichstag
reichstag
leibnizianisme, leibnizianismes
einsteinium, einsteiniums
brainstorming, brainstormings
feldspath, feldspaths
feldspathique, feldspathiques
feldspathoïde, feldspathoïdes
zemstvo, zemstvos

Fig. 3
Words for which algorithm **6.3** and the model of pronounceability are incompatible

sportsmen
sportsman
postscolaire, postscolaires
postscolairement
sounder, sounders
discounter, discounters
makhzen
jésuite, jésuites
jésuitesse, jésuitesses
jésuitique, jésuitiques
jésuitiquement
jésuitisme, jésuitismes
jésuitise, jésuitisent, jésuitises
jésuitement
déshuile, déshuilent, déshuiles

Fig. 3
(cont.)

The test agrees with intuition to indicate that the three algorithms above were given in an order of increasing adequacy.

Note that in algorithm **6.3**, the set [p t k b d g f v] is not exactly an acoustic or articulatory category of phones. If you exclude [f v] from the set to obtain the category of oral stops, or if you add [s z ʃ ʒ] to it to obtain the category of obstruents, the algorithm does not stand the test as well . . .

The following table (Fig. 4) gives the number of syllables in the case of algorithms **6.2** and **6.3**:

	6.2	**6.3**
Initial	1589	2690
Medial	1783	2140
Final	5229	3845
Isolated	3886	3886
All types	7860	6972

Fig. 4
Number of syllables in the case of algorithms **6.2** and **6.3**

The line for isolated syllables contains the number of monosyllabic words, which of course are independent of the syllabication algorithm. These figures

suggest another criterion to compare two syllabication algorithms. The better algorithm, **6.3**, leads to a smaller set of syllables. This is not surprising, but notice that the set S is still much larger than in the case of VC^* and C^*V sequences. In algorithm **6.2**, 3886 of the 7860 syllables (49%) are attested as monosyllabic words. With algorithm **6.3**, this proportion rises to 56%.

Conclusion

As a byproduct of this study, we investigated distributional constraints between several elements of a syllable, e.g. between a medial consonant cluster and the preceding vowel, or between the left and the right parts of a consonant cluster. Up to now, such distributional constraints have been easily expressed in finite automata (cf. Fig. 2). It is likely that most of the contraints at syllabic level will fit in a global automaton. If so, we will have a formal model of pronounceability.

References

Aubergé, V.–Boë, L.-J.–Lefèvre, J.P. 1988. Lexiques et groupes consonantiques. In: 17es Journées d'étude sur la parole, 55–60. Nancy.

Code typographique. 1981. Fédération nationale du personnel d'encadrement des industries polygraphiques et de la communication. 13e édition. Paris.

Désarménien, J. 1986. La division par ordinateur des mots français: application à TEX. In: Technique et science informatiques, 251–65. AFCET–Gauthier–Villars, vol. 5, no. 4. Paris.

Laporte, É. 1988. Méthodes algorithmiques et lexicales de phonétisation de textes. Doctoral thesis. University Paris 7.

Laporte, É. 1990. Le dictionnaire phonémique DELAP. In: Langue Française 87: 59–70.

Liang, F.M. 1983. Word hy-phen-a-tion by com-pu-ter. Doctoral thesis. Stanford University.

Mañas, J. 1987. Word division in Spanish. In: Communications of the ACM 30: 612–6.

Vennemann, Th. 1990. Syllable structure and syllable cut prosodies in modern standard german. In: Bertinetto, P.M.–Kenstowicz, M.–Loporcaro, M. (eds): Certamen Phonologicum II: Papers from the Cortona Phonology Meeting 1990, Torino. Rosenberg & Sellier.

Address of the author: Éric Laporte
Institut Gaspard-Monge
Université de Marne-la-Vallée
2, rue de la Butte-Verte
F–93160 Noisy-le-Grand
France

Acta Linguistica Hungarica, Vol. 41 (1–4), pp. 191–204 (1992–93)

A PROJECT FOR BILINGUAL REFERENCE CORPORA

ELISABETTA MARINAI–CAROL PETERS–EUGENIO PICCHI

Introduction

At the Institute for Computational Linguistics of the Italian National Research Council (ILC-CNR), Pisa, we are now working on the implementation of an integrated workstation that will provide functions offering on-line access to mono- and bilingual text corpora and lexical databases for in-depth search operations and analyses. Links between the text databases and the LDBs will permit lookup on detailed morphological, syntactic and semantic information in the dictionaries for any lexical item found in the texts; in addition semantic information, such as taxonomic data, derived from dictionary definition parsing operations can also be invoked to search the corpus data (for examples of this kind of text querying see Picchi–Calzolari 1986).

All the components included in the Workstation form part of the Pi-system, an open-ended modular set of tools, which has been developed to meet the various requirements of literary and linguistic text processing and analysis. The Workstation revolves around a core component constituted by the DBT, a textual database management and query system that has been implemented in different configurations to perform specific mono- and bilingual text and dictionary processing activities (see Picchi 1991). Other components are: the MLDB, a multilingual integrated lexical database system first described in Marinai *et al.* (1990) – the lexical components of the MLDB include the Italian Machine Dictionary (mainly based on the Zingarelli Italian Dictionary), the Garzanti 'Nuovo Dizionario Italiano', and the Collins Concise English/Italian Bilingual Dictionary (it is hoped to add an English LDB shortly); English/Italian Bilingual and Italian Monolingual Text Archives (for information on the Italian Reference Corpora under construction at the ILC-CNR, Pisa, see Bindi *et al.* 1991); and a Bilingual Corpus Management System which will be described in detail in section **2**. The configuration of this Workstation is shown in Fig. 1.

Akadémiai Kiadó, Budapest

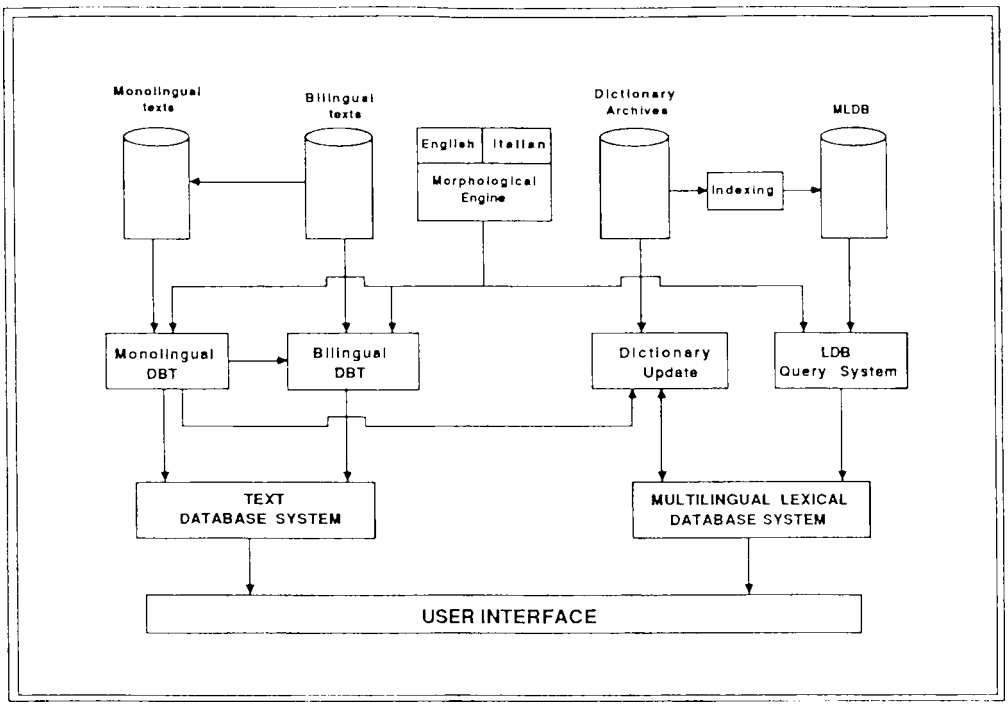


Fig. 1
The Bilingual Workstation

The Workstation can be employed in many types of activities: by the lexicographer, by the translator, by the language learner, or indeed by any user interested in exploiting to the full the possibility of being able to dynamically access, browse, and extract the different kinds of linguistic information contained in our dictionary and text databases. The user can move freely and rapidly from one component to another, using the information extracted from one to query or supplement information contained in another. The entire system is menu-driven; the user is guided in his use of each component by a standardized set of command menus and context sensitive Helps can be invoked to explain the functionality of each command.

In the paper, we will focus our attention on the bilingual components of the Workstation. Section **1** discusses the importance of bilingual reference corpora as valid sources of real-world renderings of texts written in one language (L1) in a second (L2), and illustrates their potential for exploitation in various kinds of cross-language studies. Section **2** presents a system that we have de-

veloped for the creation, management and interrogation of such corpora and finally, in section **3**, we give an example of a practical application using the system.

1. Bilingual reference corpora

In the last decade there has been a considerable growth of interest in corpus construction as language reference corpora, i.e. representative collections of texts in machine readable texts, have become widely recognised as being important sources of information, not only for lexicographical purposes but also in many other types of linguistic studies including the acquisition of knowledge for natural language processing systems.[1] This activity has been encouraged by the recent technological evolution in data storage devices and optical character readers which means that the rapid acquisition, maintenance and management of large volumes of data at relatively low costs is becoming increasingly feasible. In addition, the adoption of computerized procedures in the production of most written material nowadays (both for publication and in the office environment), means that a vast amount of texts in machine-readable form are potentially available for corpus purposes.

So far, efforts have been mainly concentrated on the construction and study of monolingual corpora, however, attention is now also being given to the creation of bilingual or parallel text archives. This interest is motivated by a growing awareness of the value of bilingual corpora as in-depth sources of documented evidence of how texts written on one language can be rendered in another, according to a number of contextual factors, such as style, register, domain, etc., and therefore of their potential for exploitation in many types of cross-language comparisons and investigations.

In the first place, such corpora can be considered as important tools in bilingual dictionary compilation, translating, and language learning activities. For example, the bilingual lexicographer can use them in much the same way as the monolingual lexicographer refers to a monolingual language corpus: as a test-bed for his intuitions; to provide reliable material for examples. While a monolingual dictionary must provide an adequate representation of the lexical system of a given language, in the bilingual the focus is on representing the complex network of relationships between the lexical system of two languages.

[1] See, for example, the success of important corpus-based dictionary projects such as COBUILD or the Trésor de la Langue Française, and the decision to set up a Network of European Corpora (NERC Consortium, 1991) to serve the various European language engineering needs, under the auspices of the European Community.

The necessity to represent a given headword on the basis of the different ways it can be rendered in the target language, and also to specify the factors which affect the choice of the TL equivalent, will influence the formulation of the SL entry and, in particular, can modify its breakdown into senses when compared to the equivalent monolingual entry. A bilingual reference corpus may well supply evidence to support the adjustment of a first SL analysis of an entry to meet the demands of the TL, and can even provide a starting point for new hypotheses on the relationships between the two languages in question.

Similarly, a bilingual corpus not only represents an important fount of inspiration for the translator, permitting him to find translations for words or expressions which are not listed in any dictionary, but could also provide important data for studies on the translation process. In fact, bilingual corpora should be considered as important repositories of data which make it possible to study many of the processes involved in transferring information, ideas and concepts from one language to another. There is much research activity at the moment focussed on the development of different kinds of procedures for syntactic and semantic tagging of texts. In particular, at Pisa, statistical methods for both POS and sense tagging are currently being studied. Once annotated bilingual texts become more easily available, methods could be developed to evaluate data extracted from them at different levels: for example, to examine changes in style and register or to investigate correspondences and differences in sentence structure and discourse organization between a target language version and its source text. Again, such studies, useful in themselves in helping us to understand and improve the work of the translator, could also permit the acquisition of data that reflects important distinctions between the two languages being considered and could help in the formulation of generalizations on the relationships between them.

Data derived from analyses on bilingual corpora could also provide valuable input for MT systems. For example, Nagao (forthcoming) stresses the importance of including detailed collocational information in the transfer dictionaries of such systems: there are many specific expressions which must be translated in a specific way in a given TL and knowledge of this sort improves the quality of an MT system greatly. To acquire such information, many collocational expressions with their translations must be accumulated and bilingual texts are important sources of such data.

1.1. Corpus design

However, the construction of a large-scale bilingual text corpus is not a task to be undertaken lightly as it implies a considerable investment of time and

resources. The goal must be a high quality corpus, sufficiently representative of the object it aims at modelling (whether the "entire" lexicon or particular sub-sets of it) and sufficiently large to provide valid data for a wide range of linguistic studies.[2] A motivated and systematic selection and collection of the texts to be included is essential. User needs must be considered at all times and exchangeability and reusability must be considered as priorities. It must be possible for a corpus constructed for research purposes to be accessed by other users of the scientific community.

Before any decisions are taken, the criteria to be adopted when assembling corpus material must be carefully evaluated. Unfortunately, no hard and fast guidelines are yet available which can be used to define the "correct" design criteria, not even for monolingual corpora. Points to be considered include the text acquisition methods to be adopted, the final size of the corpus, the definition of the sampling units, the identification of the different text types which constitute the "language" to be represented by the corpus, the proportions of each to be included (depending on the level of representativeness to be guaranteed), the time period to be covered, the labelling of samples as source or target texts (and for target texts, some indication of translation status so that the user can judge the value of the material he is handling). Consultations with language specialists and potential users of the corpus are essential during this stage in order to evaluate the correctness of the approach adopted. Another problem which has been much discussed in recent years but still remains a major difficulty is that of copyright. Although publishing companies are often ready to enter into cooperation agreements and generous in providing research institutions with access to textual material in MRF, it is not always clear what is the legal status of products derived from analyses made on such data nor whether the results can be made freely available to other members of the research community.

It is evident that the definition of standards for corpus design, management and analysis is now a priority. We expect that the work of EAGLES (Expert Advisory Group on Language Engineering Standards), a new EC initiative in the framework of the Linguistic Research and Engineering (LRE)

[2] Although corpus representativeness is a clear goal for the future, important results can still be obtained in the meantime from material which is not balanced, as long as the user is aware of the type of data he is analysing. For instance, the availability of texts, such as the Hansards transcripts of the Canadian parliamentary debates, has permitted much experimentation on the statistical processing of bilingual texts which would have been impossible without the availability of such enormous volumes of data (see, for example, work by Church and Gale 1991).

programme begun in January 1993, will be important in this area, helping also to avoid much of the actual dispersion of resources, and duplication of efforts in fragmented uncoordinated initiatives.

1.2. The test corpus

At Pisa, we have assembled a sample set of bilingual texts, selected to cover a number of different language varieties, and including extracts from scientific articles, university and school text books, magazine articles, short stories, novels, and poetry. This set of texts was collected in the first place in order to provide a test-bed for the system that we have developed for the acquisition and management of bilingual corpora, but we hope that it will provide useful empirical data to assist us in the definition of valid, more objective design criteria, which can then be used in a subsequent extension of these archives.

2. A system for bilingual corpus management

A preliminary version of the system which we have developed for the automatic construction and retrieval of parallel contexts from bilingual text archives was described by Marinai *et al.* (1991). The system has been designed to be run on sets of parallel texts—where one is the translation equivalent of the other. The user queries either of the two sets of texts and, for any form or co-occurrences of forms which can be found in the set of texts for one language, retrieves parallel contrastive contexts from the other. At the moment, the languages treated by the system are Italian and English. However, the procedures have been designed to be generalizable; given the necessary lexical components they could be transported to run on other languages.

So far most of the systems studied to manage bilingual corpora have used statistically based procedures to align the texts at the sentence level. Such programs often request the user to supply not only an SL word but also a TL candidate translation in order to construct parallel concordances. Church and Gale (1991) present a system of this type and also describe a word-based concordance tool in which the possible translations for a given word are discovered from the corpus on the basis of a pre-computed index indicating which words in one language correspond to which words in the other. Our approach to the problem is quite different. We use external evidence provided by a bilingual LDB to create links between pairs of bilingual texts on the basis of known translation equivalents, in order to retrieve "new" real world translations for words or expressions which are not attested in the dictionary.

2.1. Creating the bilingual archives

The system operates in two distinct steps. In the first stage, sets of bilingual texts are "synchronized" using morphological procedures, and a bilingual electronic dictionary (in this case derived from the Collins Concise English–Italian dictionary) in order to establish direct links between translation equivalents.

Given a new pair of bilingual texts in machine readable form, the first stage is to structure them in text database format using the DBT procedures. The texts are scanned to recognize and identify the different elements composing them. For example, word forms are distinguished from the other tokens, such as punctuation marks, numbers, line and paragraph breaks; codes are added to distinguish between full stops and abbreviation marks, between dashes and hyphens, between the different use of the apostrophe in Italian and in English, etc. This stage is simple, rapid and, once a few preliminary instructions have been given, automatic.

When a pair of texts has been memorized in DBT format, it must be input to the text "synchronization" procedure, which establishes as many links as possible between translation equivalents identified in the two texts. This procedure is totally automatic and operates as follows. Each word form in the text selected as the source text[3] is input to the morphological analyzer for that language in order to identify its base lemma, which is then searched in the bilingual LDB. All translations given for this lemma are read and input to the morphological generator for the TL; all the forms generated are then searched over the relevant search zone in the target text. If the procedure finds more than one possible base lemma for a given form, the translations for each will be read as, in the case of grammatical homography, it is quite possible that the translation equivalent does not respect the category of the source language and, in the case of lexical homography, it is presumed unlikely that the translations of the 'wrong' lemma will find a correspondence in the target text.

The algorithm requires the definition of a search zone over which it will operate when establishing the links between two texts. This zone is that area of the target text which will be searched to find the translation equivalent for a given source text form. The size of this zone is crucial as searching for

[3] For each pair of bilingual texts, we use the term Source to indicate that text, independently of the language in which it is written or whether it is an original or translated version, which is taken as input by the bilingual text "synchronization" and query procedures and on which they operate to identify or construct, respectively, for each form, the corresponding translation equivalents or contexts in the other text, denoted as Target text. The procedures can treat either of the two paired texts as Source or Target, indifferently.

translation equivalents over too large an area in the second text will greatly increase the risk of "false" matches, whereas too small a zone will mean that many "true" matches may be missed. For a new SL form, the starting point for the "synchronization" algorithm in the target text is the mid-point between the last two words in the target text for which translation equivalents have been found, and the length of the search zone is currently set at 25 word forms (examined alternately) before and after this starting point.

Articles, pronouns, prepositions, plus a small list of stop words, are excluded as being of little significance to the matching procedure and liable to create noise. When one of the translation equivalent forms is found in the searched section of the target text, a link—consisting of a physical address which locates the equivalent word in the source text, and vice versa—will be created. When no entry for a word in the source text is found in the dictionary, it may be that the form being examined is either a proper noun or a word from a highly specialised vocabulary not included in our bilingual LDB. An attempt is thus made to match such forms against any equivalent character strings in the relevant zone of the target text, ignoring the last characters to allow for morphological variations as, in the two languages in question, proper nouns and scientific terms frequently resemble each other. The matching procedure continues, word by word, to the end of the source text.

The execution of the "synchronization" procedure is rapid and totally transparent. When it is completed, the results are presented to the user in terms of the number of successful "matches" of translation equivalents between the source and target texts. The procedure will be considered to have "failed" if the number of matches is less than a given percentage of the total text. This procedure must be executed just once for each pair of bilingual texts, when they are added to the archives. An example of an output of this stage is given in Fig. 2.

2.2. Bilingual text query procedure

From Fig. 2, it can be seen that each token in both the source and target text has a value assigned to it. When a link is created between translation equivalents, the value of the SL form is associated with its TL equivalent. The values of the links are then stored together with the texts in the bilingual archives and are used by the query system in the on-line construction of parallel contexts.

For each source text word or combination of words searched by the user, the parallel contexts for the target text are constructed in real time and displayed on the screen. The source context is constructed with the searched word

Fig. 2
Examples of text "synchronization":
excerpts for a poem and a scientific article

at the centre; this word is highlighted and, during the creation of its context, any translation links to the target text associated with the other words in the context are read. If the searched word itself has an associated link, this is used to identify immediately the corresponding word in the target text, which will also be highlighted and used as the central point for the construction of the TL context. The other words that have been linked in the paired contexts can be optionally evidenced in a different colour.

When there is no directly linked TL form for the SL word being searched, then all the links for words found in the source context are used to calculate an "average" value which is used to locate the central point around which the relative TL context will be constructed. The calculation of this "average" value allows for the possibility of uneven concentrations of matched words in the contexts. The two linked forms which are closest to the point calculated as the middle of the target text context are evidenced in a different colour,

as indicators of the likely position of the translation equivalent. The user can either search for single word forms or, using the morphological generator, for all the forms of a given lemma. The archives are considered to be symmetric; either of the two languages can be selected as the "Source Language". Bilingual concordances of interest can be printed out or saved in a separate file for future reference.

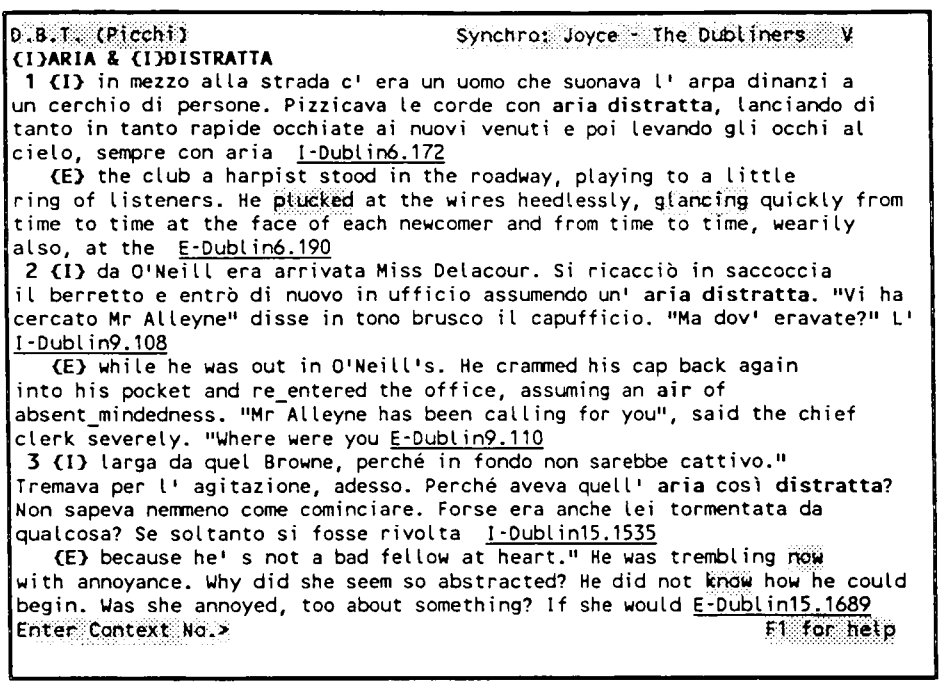


Fig. 3
Example of results of a search for all occurrences of "aria distratta" in the Bilingual Text Archives

Figure 3 gives an example of the results obtained for a query in which the Italian collocation "aria distratta" was searched in one of our bilingual sets of texts. In the figure, the words being searched and the words linked directly with them in the parallel contexts are shown in bold, whereas the indicators of the position of the translation equivalents appear in grey.

The system was first tested on the sample set of texts mentioned above, chosen to represent various types and styles of translation, in order to evaluate its performance. The feed-back from the first results enabled us to made certain

improvements to the preliminary version of the system both in the definition of the search zone for the "synchronization" algorithm and in the procedure which creates the parallel contexts when the corpus is queried. In particular, "wrong" links between falsely recognized translation equivalents which disturb context calculation are now identified and eliminated. For each direct link created, the procedure also estimates the probable value for the TL equivalent of that SL form on the basis of the other links created in the surrounding piece of text. If the estimated value differs considerably from the actual value, then the query procedure discards this particular direct link as being unreliable and recalculates the parallel contexts on the basis of those links recognised as valid.

We are now beginning to use the system to acquire larger text samples and even entire books in original and translated versions. In the next version, to be released shortly, we intend to insert a function that will provide a preliminary statistical analysis of the results. For any SL word queried, the TL equivalents found in the parallel texts will be listed in descending order of occurrence for those words for which direct links have been made, followed by the number of parallel contexts in which no direct link has been found for the searched word in the TL text. Therefore, for the occurrences of **time** in our corpus the results will appear as shown in Fig. 4, and the users can then select the particular contexts they wish to view without necessarily having to scan through them all.

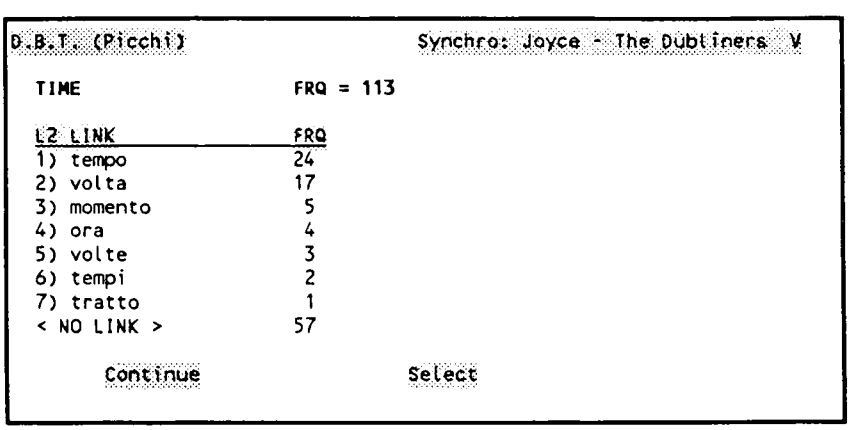


Fig. 4
Frequencies for results of a search for parallel contexts of "time" in the Bilingual Text Archives

3. Using the system

The bilingual corpus retrieval system described is currently implemented for interactive consultation, mainly to meet the needs of lexicographers, translators or language learners. It is just one of the components of the Bilingual Workstation described in the Introduction. Users of the Workstation can extract information from the bilingual archives to supplement or reassess the dictionary data contained in the LDBs. In the following, we give one example of a possible application of the system.

We have already shown in Fig. 3 how the user can query the bilingual text system to find parallel contexts for co-occurrences of words in the set of texts. In fact, one of the most important uses for a system of this type, both for the general user and for the lexicographer, is the extraction of information on the translation of bound or semi-bound expression, idioms, and frequent collocations. A printed dictionary cannot contain an extensive range of such information, owing to space factors, and the lexicographer also has the problem of deciding which information should be included, or given priority. He can use the bilingual corpus system to search relevant data by first extracting right and left sorted concordances from the set of texts for just one language, considered as a monolingual corpus,[4] in order to identify the most frequent collocates or set expressions for any given lemma. The relevant parallel contexts are then searched in order to access the appropriate translation equivalents. The bilingual lexicographer can then make an informed decision as to what should be included in his entry on the basis of the facts shown by the corpus. For example, a concordance run on our Italian set of texts in the bilingual corpus for **animo** reveals **stato/i d'animo** as one of the most frequent collocations and this expression becomes a strong candidate for inclusion in a dictionary entry.

Parallel contexts for the co-occurrence of **stato/i** and **animo** are shown in Fig. 5. The lexicographer creating an entry for **animo** can "cut" the examples that interest him from the bilingual texts, edit them, and then "paste" them into his entry using the functions provided by the bilingual dictionary editor or update system, which is one of the components of the Bilingual Workstation.

[4] In this kind of application, monolingual concordancing must be performed only on SL texts as translated texts can never give an entirely true representation of a language; they will always to some extent be dependent on or influenced by their source.

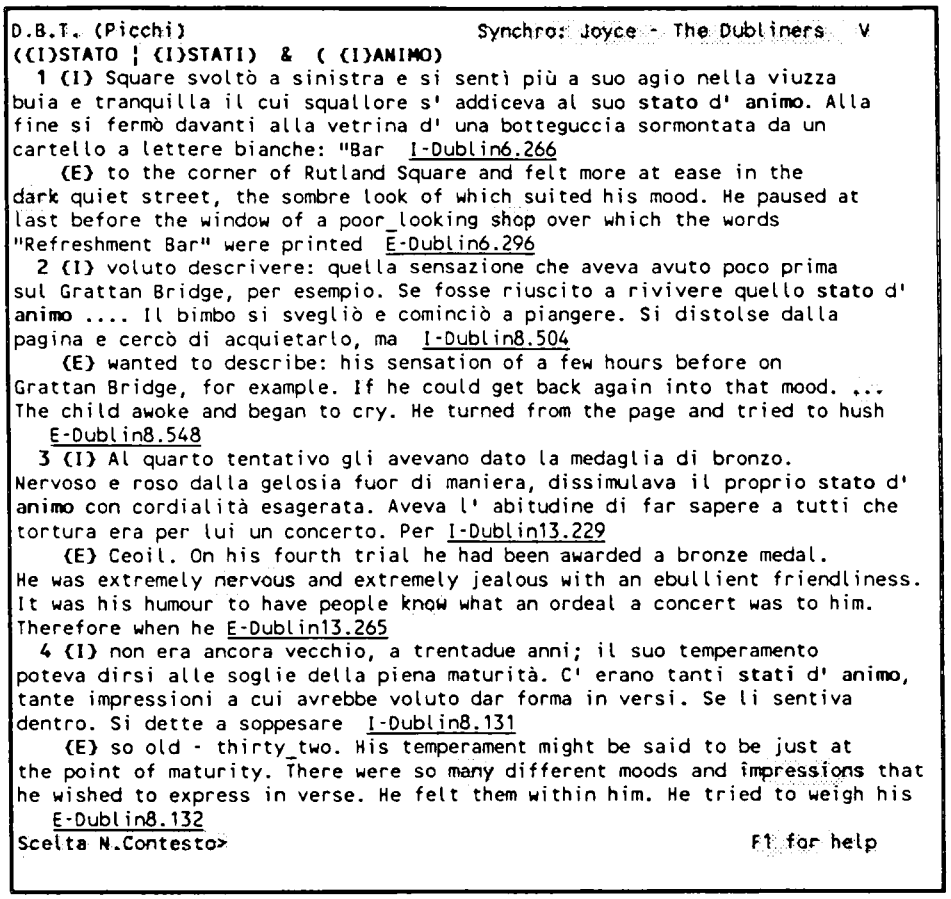


Fig. 5

Co-occurrences of **stato/i** and **animo**
in the Bilingual Text Archives

References

Bindi, R.–Monachini, M.–Orsolini, P. 1991. Italian Reference Corpus: General Information. Technical Report, ILC-CNR-TLN.

Church, K.–Gale, W. 1991. Concordances for parallel text. In: Using Corpora. Proceedings of the 7th Annual Conference of the Centre for the New Oxford English Dictionary and Text Research, 42–62. Oxford University Press, Oxford.

Marinai, E.–Peters, C.–Picchi, E. 1990. The Pisa Multilingual Lexical Database System. Esprit Basic Research Action No. 3030, Twelve Month Deliverable, ILC-ACQ-2-90, Pisa.

Marinai, E.–Peters, C.–Picchi, E. 1991. Bilingual reference corpora: a system for parallel text retrieval. In: Using Corpora. Proceedings of 7th Annual Conference of the Centre for the New Oxford English Dictionary and Text Research, 63–70. Oxford University Press, Oxford.

Nagao, M. (forthcoming.) Dictionaries for machine translation. Linguistica Computazionale. Vol. VIII.

NERC Consortium 1991. Network for European Reference Corpora. Technical Annex.

Picchi, E. 1991. D.B.T.: A textual data base system. In: Cignoni, L.–Peters, C. (eds): Computational Lexicology and Lexicography. Special Issue dedicated to Bernard Quemada. I., Linguistica Computazionale. Vol VII, 177–205.

Picchi, E.–Calzolari, N. 1986. Textual perspectives through an automatized lexicon. In: Proceedings of the XII International ALLC Conference. Slatkine, Geneva.

Dictionaries

Collins Concise English-Italian, Italian-English Dictionary. Collins, London–Glasgow, 1985.

Il Nuovo Dizionario Italiano Garzanti. Garzanti, Milano, 1984.

Zingarelli, N. 1970. Vocabolario della Lingua Italiana. Zanichelli, Bologna, 1970.

Addresses of the authors: Elisabetta Marinai and Eugenio Picchi
Istituto di Linguistica Computazionale, C.N.R.
Via della Faggiola 32
56100 Pisa
Italy

Carol Peters
Istituto di Elaborazione della Informazione, C.N.R.
Via Santa Maria 46
56100 Pisa
Italy

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 205-223 (1992-93)

COMPLEX REVISITED: REMARKS ON SOME BASIC ISSUES IN COMPUTATIONAL LEXICOGRAPHY/LEXICOLOGY*

WILLY MARTIN

1. Introduction

As the title of this paper is rather unspecified—and I am aware of the fact that even that is quite an understatement—I hasten to start by drawing up a table of contents of what I really intend to do, viz.

1. Defining the field.
2. Discussing some topical issues, within the field, in particular:
   2.1 The level of description in NLP-dictionaries.
   2.2 Re-usability of semantics in dictionaries.
   2.3 Corpus exploration techniques.
3. Rounding off with some summary conclusions.

As one may expect, in the body of this article (section **2**), I will try to express a personal view w.r.t. the topics mentioned above. Furthermore, as the title suggests, the whole is linked up with what I did talk about, two years ago, at the previous Complex-conference in Balatonfüred.

To brush up things, allow me to repeat the basic tenet and some of the main conclusions of my 1990 lecture ("Towards the construction of intelligent lexical databases"). As to the basic tenet, it was formulated as follows: "To be regarded as truly 'intelligent' the (semantic) knowledge in LDB's should—at least—be both dynamic and relational. In addition to that metaknowledge is needed."

In the end a.o. the following conclusions were reached at: "Exploration of MRD's undoubtedly is a central topic in Computational Lexicography/Lexicology.

I have a.o. tried to show that it should not be the only one. If the construction of Intelligent Lexical Data Bases is one of our primary concerns then first of all fundamental lexicological research has to be undertaken.

* Invited paper read at the 2nd Complex-conference Budapest, 4–8 October 1992.

Akadémiai Kiadó, Budapest

The critical issues are:

- research into definition models serving as prototypes of lexico-semantic knowledge;
- research into the dynamic aspects of the lexicon (in order to represent meaning (within one lexeme) more economically and more powerful)".

In what follows then I would like to re-examine some of the topics mentioned two years ago, be it from a somewhat different point of view.

2. Computational lexicography/lexicology: a delineation of the field

Taking one more look at the title of this article two items stand out. First of all the word 'basic' presupposes a framework within which topics can be ordered and evaluated and, secondly, the obvious framework 'computational lexicography' is not taken on its own here but is enlarged with, or, at least, not completely loosened from, 'computational lexicology'.

In actual fact, I hold the opinion that there is no clearcut division between computational lexicography on the one hand, and computational lexicology on the other. Of course, both have their own typical representatives, their prototypes. This gradual rather than clearcut division has to do with the fact that computational lexicography is not just and only a form of traditional lexicography with new means, lexicography in which new tools, i.e. computers, are involved. Rather it implies a change in focus. This means that, whereas the central object of traditional lexicography is the dictionary (be it in diverse forms and formats), the range of objects of computational lexicography is much wider, also implying such objects as (see Martin–Woltering 1989, 5–9):

- computer-based dictionaries
- machine-readable dictionaries
- lexical/termbanks
- machine dictionaries
- lexical databases
- artificial intelligence lexicons[1]

[1] Actually the parameters used here to differentiate between the several lexicographical objects are: physical form, formal representation of content (meaning of lexemes), any other relevant features. So AI-lexicons will show the following features:

- form :electronic
- content :formal-algorithmic
- other :contains also world-knowledge (specific knowledge about the State-of-Affairs in the domain treated)

The more computational lexicography is putting lexical databases instead of computer-based dictionaries in the centre of its interest, the more the boundary between computational lexicography and computational lexicology is disappearing, both converging towards a common object which we call the computational lexicon.

To illustrate more clearly what I mean the following quotation may be of use: "We would like to suppose that every new dictionary that is published nowadays has been derived from an underlying database (which can, of course, be more or less sophisticated). Furthermore we suppose that different types of users are in need of different types of dictionaries. [...] This does not imply, however, that one has to build up a completely separate database for each type of dictionary. *For several reasons it is preferable to set up one 'subjacent', fundamental database which is not user-oriented, and to derive from it as many user-oriented front-end databases as there are types of dictionaries.*

Editing, updating and further completion of the bare lexical data are activities which—in a certain sense—should be unrelated to the final products a publisher may have in mind. They concern the fundamental, non-user-oriented database. Front-end databases, on the contrary, are typically product-oriented. They contain specfic selections which depend entirely on the needs of the users for whom the dictionaries involved are intended" (Martin–Al 1990, 393).

The idea of an underlying lexical (knowledge) source serving different applications is very clearly visualized in Fig. 1 to be found in Heid (1991).

At the centre we find the underlying non-user-oriented lexical database, whereas to the right of it several application-oriented lexicons are to be found, derived from the central, underlying one by means of extraction devices.

At the same time this picture makes clear that, although computational lexicography, as a rule, is user-oriented and so linked up with applications, the latter cannot be but related to representation and acquisition aspects as well. In its turn computational lexicology although being primarily interested in the computational lexicon as an underlying database cannot escape the confrontation of acquiring and applying the knowledge gathered. Given this situation we will, in what follows, deal with topics which refer to these three basic aspects of computational lexicography/lexicology viz.

- the application
- the representation
- and the acquisition of computational lexicons.

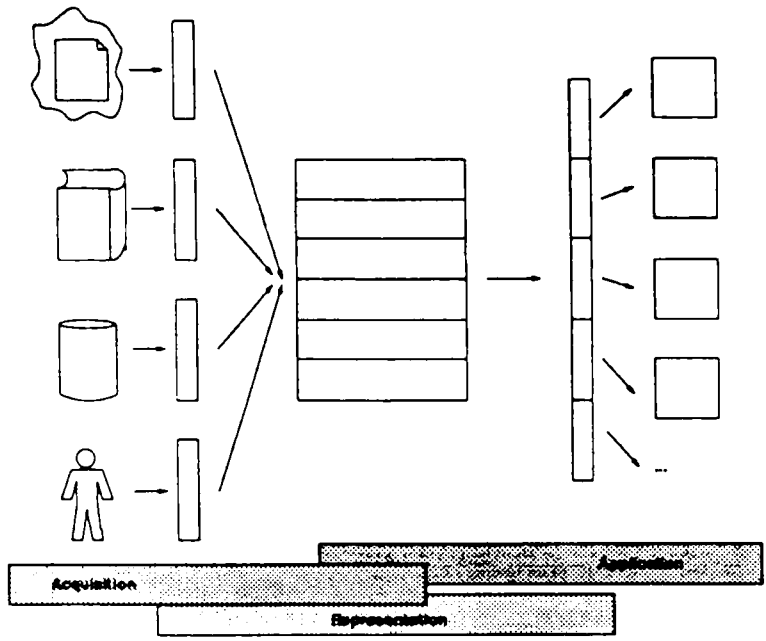


Fig. 1
A schematic overview of a general reuse scenario

3. Applications and the level of description

If computational lexicology/lexicography is primarily concerned with computer lexicons in the sense of lexical components of multi-functional, multi-purpose NLP-systems, then one can expect that this variety of applications will call for a *flexible representation and description.*

One report which renders this issue very clearly is the *Eurotra-7 Study*: a feasibility and project definition study on the re-usability of lexical and terminological resources in computerized applications carried out under the auspices of the EC by a European consortium of academic and industrial partners (see Heid–McNaught 1991).

On page 72 of the above-mentioned report one can read the following: "Within descriptive linguistics, different theories and descriptive models are basically interested in the same phenomena, but they classify the phenomena in different ways, using different parameters to delimit subjects of the set of phenomena they are faced with.

Such classifications of the individual objects of an observational domain allow for different, even for a priori incompatible generalizations." (Heid–McNaught 1991).

One of the basic tasks then of the computational lexicographer/lexicologist is that, given a set of relevant phenomena captured by a certain theory, he should not lose sight of the fact that the lexicon should "contain a representation of the observable linguistic facts used for the definition of theory-specific concepts" (also see Heid–McNaught 1991, 71). Let me illustrate the above with an example taken from my own practice.

Some years ago when being interested in lexical or word frequency, I soon observed that, contrary to what was commonly held at the time, frequency was not simply to be put on a par with the actual number of occurrences a lexeme had in a corpus. Instead of that I started from what I called the frequential lexical communicative competence native speakers possess. With this I meant that a.o. native speakers could come to such statements as: very frequent, frequent, more or less neutral w.r.t. frequency (= not frequent, not infrequent either), unusual, very rare etc. In other words, I considered word frequency values to function at an ordinal, not at a ratio scale, implying that native speakers could de-contextualize frequency and so were able to observe the frequential similarity between e.g. *bacon* and *cheese* or *June* and *November* although the frequential observational data of the words just mentioned may differ.[2] Moreover the lexical frequency values, in my opinion, reflected a consistent system, be it one dependent on and concomitant with other linguistic-communicative systems.

Given this system,[3] predictions could be made meaning that, given the frequency of item A, that of B could be predicted (see Figs. 2 and 3 taken from Martin 1988b).

[2] There are, of course, plenty of examples of so-called lexical frequency variation. So e.g. *bacon* has frequency = 27 in a corpus of 1 million wordtokens (The Leuven Drama Corpus) and *cheese* has frequency = 52 in the same corpus. The same goes for *June* (f = 93) and *November* (f = 50) in e.g. the LOB-Corpus.

[3] The system transforms so-called 'objective' frequency values into objective-subjective ones bringing them (more) in accordance with native speakers' intuitions by putting them on an ordinal, instead of on a ratio scale (for an explanation see Martin 1983; for a full application see Martin–Tops 1984, 1989). General predictions such as frequential equivalence relations between co-hyponyms, all other things being equal, can be observed to make sense now (see Figs. 2–3 'days of week', 'months of year'), such as systematic explanations for similarities/differences can be given (see Table 1: 'animal + typical sound': if verbs are restricted to nouns then, given the frequency for the nouns to be the same, that of the corresponding verbs will belong to the same, lower, frequency class, except when synonyms for the verbal expressions are found).

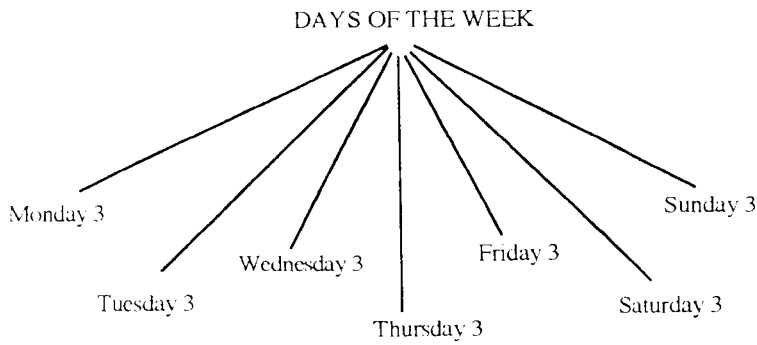


Fig. 2
Days of the week: similar-taxonomical relationship

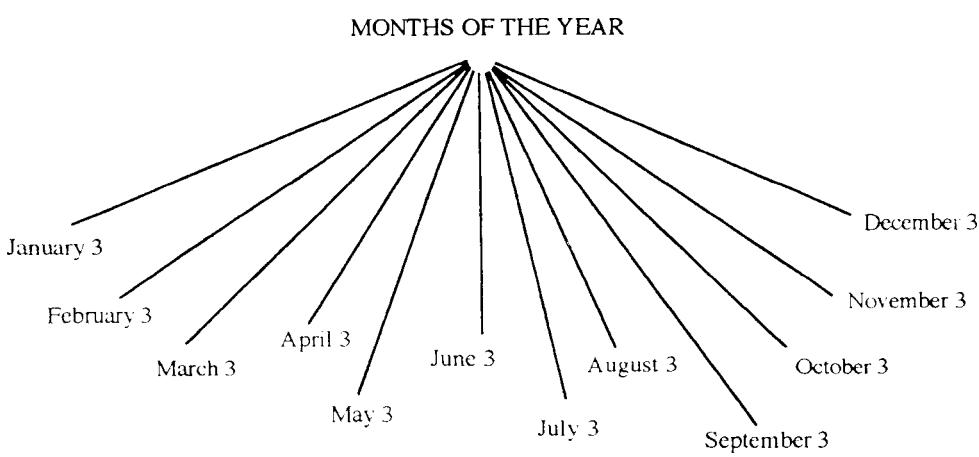


Fig. 3
Months of the year: similar taxonomical relationships

In the same way also 'explanations' could be provided for. See e.g. the difference between 'bark' and 'whinny' 'neigh' in Table 1 underneath (from Martin 1988b).

Table 1

Name of animals + typical sound they produce			
ANIMAL		SOUND	
bee	2	buzz	2
cat	3	miaow	1
		mew	1
cock	3	crow	1
cow	3	moo	
dog	3	bark	2
donkey	2	bray	1
duck	2	quack	1
goat	2	bleat	1
goose	2	honk	1
hen	3	cackle	1
horse	3	neigh	1
		whinny	1
pig	3	grunt	2
pigeon	2	coo	1
sheep	3	bleat	1

What I wanted to make clear with this digression is that, as a rule, there is a gap between the observational layer and the (theory-dependent) descriptive layer as Fig. 4 shows.

In other words, though interesting the framework I have sketched about lexical frequency may be, in order to function maximally (in as many as possible applications, and with a maximum of compatibility with other systems) the lexicon should at least *also* contain those data that are used to make the distinctions the theory considers to be relevant. W.r.t. lexical frequency data

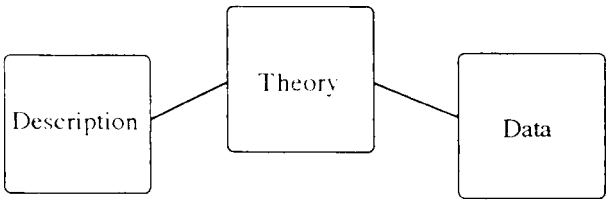


Fig. 4

this would imply the actual occurrence of the data to be observed over several informants (corpora) chosen according to relevant linguistic parameters.[4]

4. Re-usability of dictionary definitions

Traditionally in printed monolingual dictionaries one can find information w.r.t. the meaning of words scattered over a variety of categories (definitions, examples, grammatical and pragmatic data, combinatorics etc.). Of course, par excellence, definition[5] is the place to look for when in need of semantic information. During the last decade much effort has been spent to make definitional information more accessible to NLP-systems. Apparently, processing language, be it actively or passively, by humans or by machines, presupposes knowledge and, at least part of it seems to be relational and to reside in the lexicon. A.o. knowing that 'beard' is to/can be taken as an attribute of 'man' (see Cobuild i.v. 'beard': 'A man's beard is the hair that grows on his chin and cheeks') makes the sentence 'I saw the man with the beard' univocal, and, in this sense, helps to understand it, i.e. not to misunderstand it. The same goes for generating language: Knowing what 'mew' means (see Cobuild i.v. 'mew': 'When a cat mews, it makes a soft high-pitched noise') can, given the question: 'Have you seen my cat?', lead to the following answer: 'No, but I have heard something mewing nearby'. Given this role and function of lexical semantic knowledge within language processing systems, it may be worthwhile to take a look at

- dictionary definitions as an input
- strategies to parse and output aimed at

and to spend a couple of remarks on each of these issues.

[4] For more information see Martin (1983).

[5] For easiness' sake we will in this paper make use of the term 'definition' when actually only part of it, viz. the 'definiens', is meant.

4.1. Dictionary definitions as input

Although I greatly admire the performance, the skills and the intuitions of many lexicographers, I think that many definitions in dictionaries are considered useful mainly because of the fact that a) a good understander only needs half a word and b) that 'halving' need not necessarily be done according to a consistent all-encompassing system. To illustrate what I mean I will give a couple of examples taken from a dictionary I personally appreciate very much, viz. the 'Basiswoordenboek Nederlands' (Basic Dictionary of Dutch).

Under 'mus' (sparrow) e.g. one finds[6] in this dictionary: 'a grey-brown little bird', from this, one could conclude that, w.r.t. a relative notion as 'size', the comparison is made at the level of a prototypical genus proximum (bird), yet, as no specification is given under 'vogel' (bird) w.r.t. prototypical size nor w.r.t. a prototype as such, it remains up to the user to specify size and/or prototypicality w.r.t. such birds as: 'merel' (blackbird), 'zwaluw' (swallow), or 'ekster' (magpie) etc.

A second example can illustrate the lack of systematicity found in many definitions. If we take again the same dictionary as information source the definition for 'neushoorn' (rhinoceros) specifies the habitat for the animal in question, viz. Africa and Asia. If however one takes a look at 'olifant' (elephant), then no habitat is specified. With 'struisvogel' (ostrich) on the other hand it is (Africa), with 'krokodil' (crocodile) it is not etc.

These examples can make clear that re-using 'human' definitions in NLP-systems cannot simply mean to take what is given for granted but that, if need arises, one will have to expand, to specify and to modify what is presented.

4.2. Parsing techniques

As will have become clear from the preceding, analytical lexicographical definitions typically are relational knowledge structures: starting from the meaning of the definiendum they link it up, relate it to, the meaning of one or more meanings in the definiens. This idea of the relational lexicon, in the sense of one in which word meanings are linked up by means of lexical semantic relations, and in so doing form word meaning-relation-word meaning triples-, has been the starting point for the work of Raoul Smith, Martha Evens and their collaborators (see Evens 1988). In particular they have been interested in the way lexical semantic relations are expressed *lexically* in definitions by means of so-called *defining formulae*, i.e. certain words or phrases like 'characterized by', 'a state of', 'a group of' etc. Parsing of definitions for them basically comes

[6] The Dutch explanations have been translated into English.

down to integrate strings such as defining formulae into a (mainly) context-free rewriting string grammar. The parse tree (Fig. 5) can give an idea of their method of working (taken from Alshwede–Evens 1988).

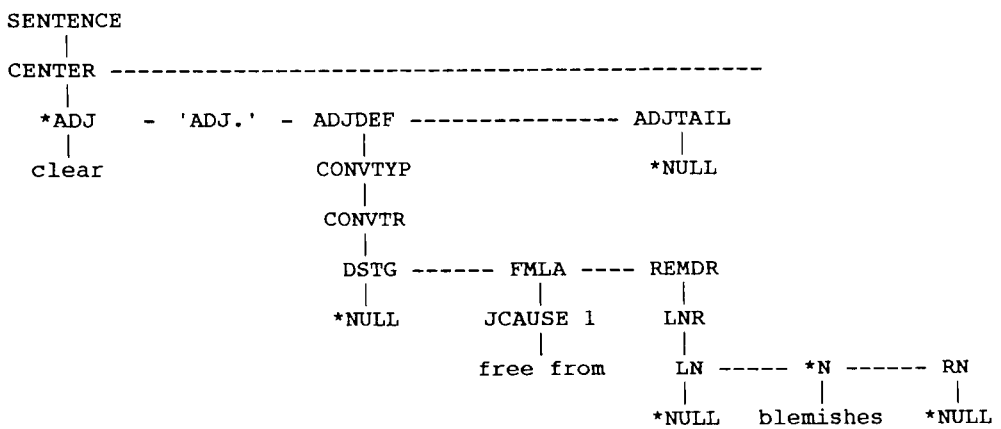


Fig. 5

Actually, Evens *et al.* set the pace for what I will call *syntax driven* parsing of definitions such as to be found in the work of Alshawi (1989) and Montemagni–Vanderwende (1992) for instance.

In these and analogous approaches "in order to achieve reliable semantic accuracy in the extraction process, [..] the definition text should be parsed [syntactically]" (Montemagni 1992, 265).

So in order to find out that the fillers of the purpose slot for laboratory are 'study', 'testing' and 'analysis', one has first to parse the definition 'a place equipped for experimental study in a science or for testing and analysis' in a structure such as shown in Fig. 6 (based on Montemagni–Vanderwende 1992).

Then a rule such as 'if the PP with 'for' is not a post-modifier of a verb 'used', then a PURPOSE relation between the definiendum and the head(s) of the PP can be hypothesized if the nearest noun that the PP post-modifies is the genus term', can be applied (Montemagni–Vanderwende 1992, 528).

It has always struck me that, when talking about parsing of definitions, lexical and/or syntactic structures were discussed first, conceptual ones on the other hand, came last, if at all.

LABORATORY
PURPOSE 'STUDY,"TESTING,"ANALYSIS'

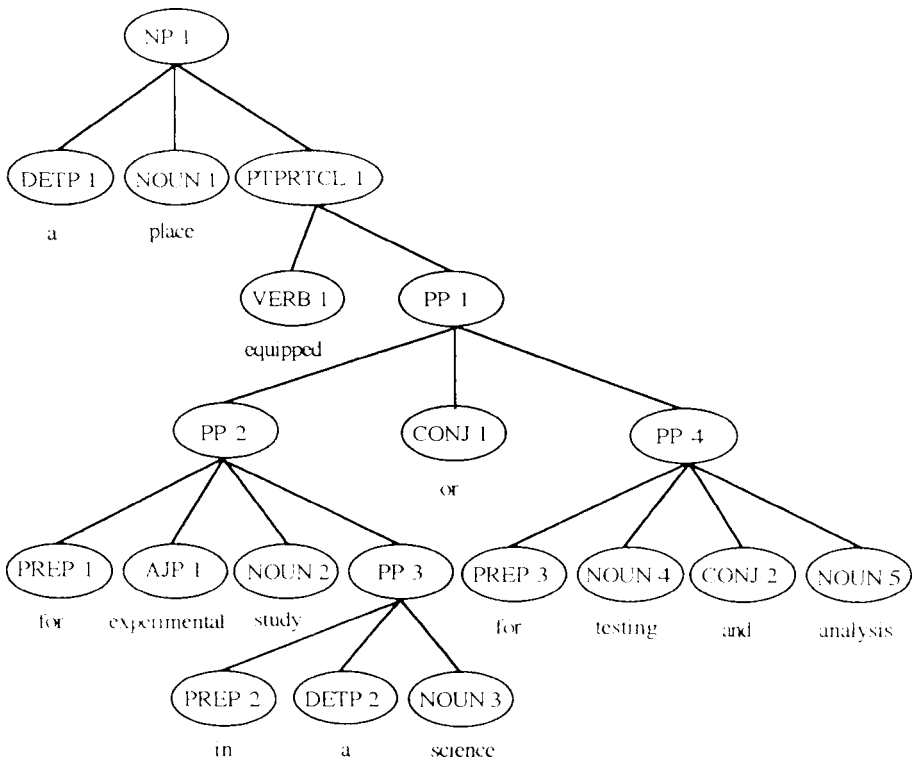


Fig. 6
Semantic frame for "laboratory" and the parse from which it was derived

In my opinion this should just be the other way round. In other words, when we want to parse verbalizations of concepts[7] we should know about the conceptual system we start from and ultimately want to map or represent into.

In order to get a better insight into what I will call a *concept driven approach* to parsing, a small project was carried out a year or so ago. In it a small team of researchers consisting of M. Reedijk, E. ten Pas, L. Willekens

[7] Of course, not all words have a mere conceptual meaning, yet, for most of them, conceptual meaning is the central meaning aspect, see Neubert (1978) and Martin (1988b) in this respect.

and myself developed a prototype parser meant to tackle the definitions of diseases (see Reedijk 1991; Martin 1992a, b)

In what follows I will not give you the full details of the system in question, instead of that I will try to give you a general idea of the approach followed, of its requirements, possibilities and its limits.

1. First of all a conceptual system was constructed. In it the concept 'disease' was given a central place and its relation with other concepts clarified. The system was only partial in that concepts related to disease in their turn were not always fully specified.

2. To construct the system we mainly made use of two information sources, viz.
   - definitions of disease-instantiations (e.g. cancer, hepatitis, aids etc.)
   - terminological combinations with (the) disease concept(s) (e.g. liver disease, breast cancer, hepatitis etc.).

Because of lack of time we dismissed other possible sources such as experts, metaphors (diseases attack, strike people down, people have to fight against them, diseases can be malignant, etc.) etc.

3. The following schema (Fig. 7) should give you a better idea of what the conceptual system looked like (see Martin 1992a).
4. Looking at the conceptual system above it should become clear now what kind of representational structure we ultimately would like to come to, viz. one in which concepts for the second argument in a relational triple would become specified. To understand what is meant compare the general unspecified nosology concept to the one specified by the information contained in the definition for 'asthma'.

```
nos_concept
    g_affects (nos, macro, micro funct, embryo)
    o_affects (nos, organism)
    caused_by (nos, etiology)
    has_systom (nos, finding)
    transmitted_by (nos, trans)
    has_qual (nos, qual)
```

asthma: "a respiratory disorder, often of allergic origin, characterized by difficulty in breathing, wheezing, and a sense of constriction in the chest"

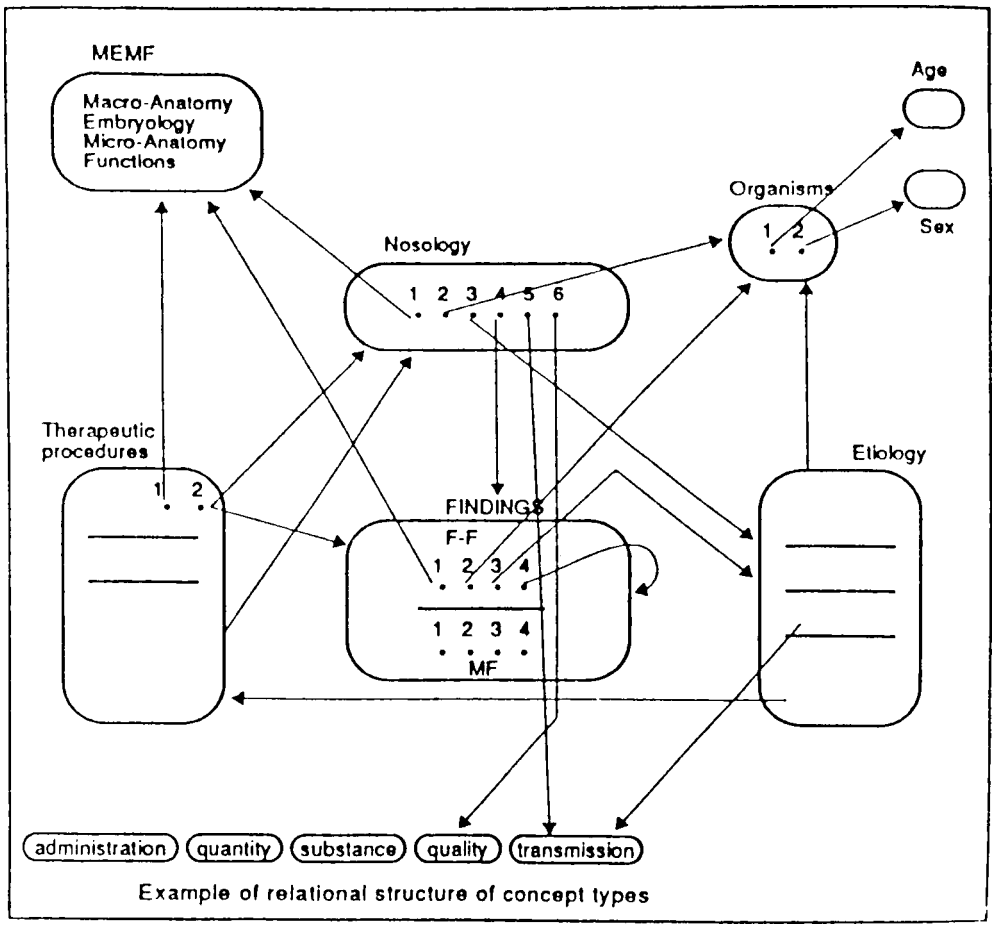


Fig. 7
Part of conceptual system 'diseases'

```
asthma:
[disease   caused_by       allergy]
[disease   g_affects       respiration]
[disease   has_symptom     constriction]
[disease   has_symptom     wheezing]
[disease   has_symptom     difficulty]
               [constriction      g_affects   chest]
               [difficulty        g_affects   breathing]
```

5. It should be obvious that, given the fact that we want to link a concept (the definiendum) to other concepts (as contained in the definiens), the system should have access to conceptual knowledge and be able to do some conceptual computation. As such therefore the lexicon will contain lexemes which are to be regarded as three-place predicates consisting of the actual lexeme, its concept type and its word category. So e.g. (aids, concept, (nosology-concept, aids, [u,u,u,u,u,u]), n)).

6. It is not the place here to elaborate on the conceptual computation. We refer to Martin (1992a, b) in this respect. Here it may suffice to state that with a minimum of syntax and with fairly straightforward conceptual calculation very good results could be obtained in which classical problems of ambiguity can be solved in an elegant way.[8]

7. The main question is however whether the techniques advocated here can easily be used when no longer dealing with terms. As is well known, terms show a much stronger resemblance to concepts as there is, as a rule, a one-to-one relationship between term and concept. Secondly one should keep in mind that definitions of terms as compared to definitions of general language words, are rather stipulations of meaning than hypotheses about meaning (see also Wiegand 1984, 20). In this respect it will be undoubtedly much more difficult to define concept types for general lexical items, yet, in my opinion, it is not at all unfeasible[9] and, moreover, looking at what to represent in the computational lexicon, conceptual knowledge is an issue one cannot escape.

5. Corpus-exploitation and the knowledge-acquisition issue

Of course, what has been stated in the preceding section w.r.t. the parsing of definitions can also be looked at from the angle of lexical knowledge-acquisition. Yet, the remarks I will give on the latter issue will be based rather

[8] A.o cases of scope of co-ordination and PP-attachment can be more easily solved now in that conceptual knowledge can be used as a checkpoint. So e.g. in the definition of 'rheumatoid arthritis': "a chronic disease of the musculo-skeletal system, characterized by inflammation and swelling of the joints, muscle weakness, and fatigue" 'joints' will be correctly linked to both 'inflammation' and 'swelling'.

[9] The crux is not so much to define the 'slots' of the type, as to find the specific fillers for them, given the definiens. Yet having gained already some experience within general language domains such as professions/locations/vehicles and the like, does not give rise to rejection of the idea.

on corpora than on definitions. Indeed, corpora can be of great help in the study of the lexicon in general and in that of the computational lexicon in particular. Actually, in my opinion, the exploration of computer corpora for lexicographical purposes or in other words, the development of algorithmic empirical analysis techniques is or should be one of the primary pre-occupations of the computational lexicographer/lexicologist. With this I do not mean that the latter should be involved in the making of concordances, KWIC-indexes or the like, rather I expect from a computational lexicographer/lexicologist that, given the nature of general language words (cf. supra), he can e.g. make hypotheses about he meaning a certain word gets within a linguistic community and tries to develop procedures to automatically test his hypotheses. Interesting hypotheses are a.o., in my opinion, those with which to catch the potentiality of the meaning of words. Let me make clear what I mean by giving an example. Looking at Van Dale's Dictionary of Contemporary Dutch i.v. 'ei' (egg) one finds the following definitions (I translate and represent them in a somewhat shortened form):

1. female reproductive cell from which after fertilization a new individual develops
2. a germ as in 1 from birds, more in particular one from hens (meant as food)
3. something which has the form of an egg
4. softy.

From a contemporary dictionary of Dutch I would expect something quite different. Actually, I do not expect an undifferentiated enumeration of meanings, rather I would think of 'ei/egg' as something having a basic meaning from which other meanings are/can be derived. Using the terminology from the preceding section I could also say that with 'ei/egg' there corresponds a basic or cognitive type[10] from which several tokens (instantiations) may be derived. For the case at stake we could imagine this type to be, given our culture and time, 'that what is laid by hens, which has a characteristic form and what we eat'. From this we can derive

[10] See Martin (1990), therein Neubert (1990) is quoted with the term 'cognitive prototype'.

a) the comparable product with the same form from 1) other birds and 2) from animals other than birds

b) any object with the form of an egg (in its basic meaning)[11]

c) in a specialized biological sense: the biological equivalent of egg.

Not to be derived (at least literally) is then 'a softie'.

As one will have observed, for me a corpus is not a discovery procedure per se. Instead of that, I expect from a computational lexicographer/lexicologist that he/she knows how to explore corpora and tries to computerize these explorations. Generalizing from the above this would mean that one considered the producers, the users, the usages/functions and the formal characteristics as the most important semantic actants of 'ei/egg' (see Nakhimovsky 1990, 6 ff.).

In the exploration of a corpus as a means to acquire knowledgde, it is, in my opinion, therefore important that

a. by way of hypothesis, the semantic actants are defined; ideally this could be done via a supertype which 'generates' the possible semantic actants which are at stake (e.g. those which are expected in the case of 'animal products taken as food' such as 'honey', 'milk' and 'egg';

b. given certain expected actants, one tries to develop algorithms to find the corresponding syntactic expressions for them (e.g. in many cases the subject of a sentence will be the logical subject or first semantic actant of the predicate investigated); in so doing the starting hypothesis (see a.) will be tested/specified;

c. special attention should be paid to (syntactic) structures (such as co-ordination e.g.) which do not refer to constituent actants but to synonyms, co-hyponyms etc. of the lexemes under investigation;[12]

d. one should not lose sight of the fact that

   1) in order to come to an exploration as sketched in a. through c. in as automatic as possible a way, the corpus should at least be syntactically tagged;

[11] Notice that it was unclear in the dictionary mentioned (see 3rd meaning) which meaning the reference was made to.

[12] So e.g. the fact that 'desire' is co-ordinated with other intense emotions such as 'anger' ('anger and desire fought in him') or that 'honey' is combined with 'marmalade' (as in 'different marmalades and honey') is revealing for the semantic nature of 'desire', resp. 'honey'.

2) not all words/actants behave in the same way; in order to find information about an actant 'result' in the case of a predicate 'destroy' one will need more context than the one needed to specify the 'theme' of such an action;
3) interpretation of the data gathered still remains necessary.

6. Concluding remarks

In this paper I have tried to make clear that

- the difference between computational lexicography and computational lexicology is not a clearcut categorial, but a gradual one; both having the computational lexicon as their common object of interest, be it that the former looks at it more from a 'peripheral' perspective (and so 'starts' by paying more attention to empirical basis and applicative aspects), whereas the latter takes a more 'central' view on it, studying the computational lexicon as a system, a structure, of its own, so to apply it later on in as many applications as possible;
- as computational lexicons are to be regarded as flexible multi-purpose knowledge sources, the theory specific concepts they make use of, should be made explicit at the observational level;
- dictionary definitions should be parsed not within a syntactic-semasiological framework but within a conceptual-onomasiological one;
- corpora should not be used first and foremost as discovery procedures for meaning and/or combinatorics (as in many statistical approaches), but within a framework of hypothesis testing; one of the main tasks of computational lexicography/lexicology then being the algorithmization of the generation and testing of hypotheses.

Confronting this with what I have stated in the beginning and what I did state two years ago w.r.t. the exploration of the MRD's, I can only repeat that, in my opinion, we can not simply stick to what is to be found in knowledge sources such as 'human' dictionaries and /or corpora, in any case, we have to supercede the data found there to integrate them into a system which is at the same time both less rigid (and so more general) and more specific (and so more explicit).

References

Alshawi, H. 1989. Analysing the dictionary definitions. In: Borguraev, B.–Briscoe, T. (eds): Computational Lexicography for NLP, 153–69. Longman, London–New York,

Alshwede, T.–Evens, M. 1988. Generating a relational lexicon from a machine readable dictionary. In: IJL 1: 214–37.

Evens, M.W. (ed.). 1988. Relational Models of the Lexicon: Representing Knowledge in Semantic Networks. Cambridge Unversity Press, Cambridge.

Heid, U. 1991. A short report on the Eurotra-7 Study. Ms. Stuttgart.

Heid, U.–McNaught, J. 1991. Eurotra-7 Study. Stuttgart.

Huijgen, M.–Verburg, M. 1987. Basiswoordenboek van de Nederlandse Taal. Van Dale Lexicografie, Utrecht–Antwerpen.

Martin, W. 1983. The construction of a basic vocabulary: An objective-subjective approach. In: Linguistica Computazionale 3: 183–97.

Martin, W. 1988a. Variation in lexical frequency. In: van Reenen, P.– van Reenen-Stein, K. (eds): Distributions spatiales et temporelles, 139–52. Benjamins, Amsterdam.

Martin, W. 1988b. Een kwestie van woorden [A matter of words]. Inaugural lecture. Free University of Amsterdam.

Martin, W. 1990a. On the organization of semantic data in passive bilingual dictionaries. In: Euralex '90 Proceedings, 193–201. Bibliograf, Barcelona.

Martin, W. 1990b. Towards the construction of intelligent lexical databases. Ms. Paper read at the first Complex-conference. Balatonfüred, October 1990.

Martin, W. 1992a. On the parsing of definitions. In: Tommola, H. *et al.* (eds): Euralex '92, Proceedings I–II, 247–56. University of Tampere, Tampere.

Martin, W. 1992b. Concept-oriented parsing of definitions. In: Coling-92, vol. III, 988–92.

Martin, W.–Al, B. 1990, User-orientation in dictionaries: 9 propositions. In: Magay, T.–Zigány, J. (eds): Budalex '88 Proceedings, 393–9. Akadémiai Kiadó, Budapest.

Martin, W.–Tops, G. (eds.). 1984, 1989. Van Dale Groot Woordenboek Engels-Nederlands. Van Dale Lexicografie, Utrecht–Antwerpen.

Martin, W.–Woltering, M. 1989. Basic Issues in Computational Lexicography. (Research paper on the state-of-the-art and the prospects of dictionary technology). Van Dale Lexicografie, Utrecht.

Montemagni, S. 1992. Tailoring a broad coverage grammar for the analysis of dictionary definitions. In: Tommola, H. *et al.* (eds): Euralex '92, Proceedings I–II, 265–75. University of Tampere, Tampere.

Montemagni, S.–Vanderwende, L. 1992. Structural patterns vs. string patterns for extracting semantic information from dictionaries. In: Coling '92, vol. II, 546–52.

Nakhimovsky, A. 1990. Word meaning and syntactic structure: some comparative notes. In: Steele, J. (ed.): Meaning-Text Theory, 3–17. University of Ottawa Press, Ottawa.

Neubert, A. 1978. Kinds of lexical meaning. In: ZAA 26: 241–6.

Neubert, A. 1990. Fact and fiction of the bilingual dictionary. In: Euralex '90 Proceedings, 29–42. Bibliograf, Barcelona.

Reedijk, M. 1991. Een conceptuele parser voor definities van medische termen [A conceptual parser for definitions of medical terms]. Ms. Free University of Amsterdam.

Sinclair, J. (ed.). 1987. Collins-Cobuild English Language Dictionary. Collins, London–Glasgow.

Sterkenburg, P. van–Pijnenburg, W. (eds.). 1984. Van Dale Groot Woordenboek Hedendaags Nederlands. Van Dale Lexicografie, Utrecht–Antwerpen.

Wiegand, H.E. 1984. On the structure and contents of a general theory of terminology. In: Hartmann, R.R.K. (ed.): Lexeter '83 Proceedings, 13–30. Max Niemeyer, Tübingen.

Address of the author: Willy Martin
Vrije Universiteit, Amsterdam
Instituut voor Lexicologie
De Boelelaan 1105
1081 HV Amsterdam
Holland

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 225-233 (1992-93)

COMPUTATIONAL LEXICOGRAPHY: A QUERY SYSTEM FOR TEXT CORPORA

MONICA MONACHINI-EUGENIO PICCHI

1. Introduction

The work presented in this paper has been inspired by two important trends which have emerged in the last few years: the considerable attention being paid by the scientific community to large, shareable and interchangeable linguistic resources and the wide-spread increasing interest in large text corpora (Calzolari–Zampolli 1990). Text corpora are needed by a growing number of academic and industrial users for many kinds of purposes, both scientific and commercial (NERC Consortium, 1992). To mention just a few examples, they are useful in linguistic sectors, in NLP activities, in psycho-linguistics, in lexicography and lexicology, in language teaching; they are considered one of the most valuable tools within the language industry community. A detailed analysis of users and uses of the Italian Reference Corpus is presented in Monachini (1992).

The collection of enormous quantities of linguistic data and in particular of text corpora of million of words has software implications (Sinclair 1991). The creation of tools for data analysis and classification, and the storage of the results of these operations are economically relevant tasks.

There is thus a need for systems which, on the one hand, offer rapid, direct access to all the different elements contained in a text, and also to the results of analyses performed on the text and, on the other, tools which permit the information to be reused[1] by applications for knowledge extraction from corpora (Bindi *et al.* 1991). The more powerful and interactively oriented the tool, the better.

At our Institute, procedures which operate on textual and lexical data have been studied extensively over the last years. The result has been the Pi-system, a set of tools developed to meet the needs of a wide range of literary

[1] The notion of "reusability" has become central in research on large linguistic resources (Calzolari–Zampolli 1990). In the collection of million of words it is important to know how to retrieve information and what to do with it if we do.

and linguistic text processing activities. The kernel of this system is the textual database management system, DBT (Picchi 1983 or 1991).

The system presented in this paper is a component of the above Pi-system and has been designed to permit the user to access and consult text corpora interactively, according to the interests of the user. Particular procedures have been implemented so that many different types of linguistic information stored explicitly and implicitly in the texts can be searched and retrieved.

In this way, vast quantities of valuable linguistic information, otherwise inaccessible, can be consulted.

The work presented here is being carried out within the project "Network of European Reference Corpora–NERC" (NERC Consortium 1991), whose main goal is to evaluate the feasibility of common criteria for composition, standards for coding, strategies, methodologies and tools as far as corpora are concerned (Zampolli 1990).

2. A database for tagged corpora

2.1. Tags, tagging and tagged corpora

Among the possible annotation practices, i.e. the practice of adding interpretative linguistic information to a corpus at various levels by some kind of coding (Leech, forthcoming), grammatical tagging is the most known and familiar because it can be performed largely automatically (Garside *et al.* 1987; Marcus–Santorini, forthcoming).

Tagging classifies morphosyntactically each word-form in a text, labelling it with a tag, i.e. a simple or complex code which encodes information on the part of speech (PoS) and the morphological features.

Tagged corpora are useful, in general, in NLP, where the knowledge derived from corpora is used to improve rule-based parsers or to provide a basis for parser with a statistical approach.

They play a crucial role in studies on the lexicon, where word class and meaning are often dependent one on the other (Sinclair 1991 and forthcoming).

Corpora can integrate the lack of certain types of information in paper, machine readable dictionaries, and in computational lexica (Bindi *et al.*, forthcoming). However, tagged corpora must not only be considered as a step towards parsing, they are important in themselves as repositories of linguistic phenomena, for example, real sequences of grammatical categories. In lexicography, they provide documented evidence of the real usage of a word and of all the related linguistic facts which can be usefully recorded in a corpus-based dictionary entry.

2.2. Handling tagged corpora

The tool we are presenting in this paper has been conceived as a computational aid for corpus users enabling them to handle tagged corpora accessing interactively the entire text, the single words and all the various levels of linguistic annotation: the tag attached to each word form, the word+tag pair, and, when the text is lemmatised, any lemma linked to a word form or the lemma+form set.

The query system is menu-driven and user-friendly. The user can easily formulate simple and complex queries without prior knowledge of the command language: during a query session, the screen presents a menu of all the commands which can be used; commands are entered by selecting them from the menu or using their key letters which are highlighted. In response to a query, the system displays the context(s) in which the object of the query is instantiated.

2.3. Structure of the database

The tagged corpus data to be acquired by the system are structured in DBT format: this implies an analysis of the text to recognize the different elements composing it and its transformation into DBT structure. For a full description of procedures for the DBT acquisition and structuring of texts (Picchi 1991).

The system operates on textual data at various levels:

- the *text*, seen as coherent string/span of written natural language
- the *tag*, i.e. a simple or complex code which encodes information on the part of speech (PoS) and the morphological features, associated with each word form
- the *lemma*, which connects all instances of word forms of the same lemma (the lemma can be seen either as equivalent of the dictionary headword without solving the polysemy, or at the word sense level where polysemy has been disambiguated).

The data is stored in archives structured as follows:

Base level	the archive contains a structured copy of the text in DBT format. If the text is tagged, each word in the text is stored together with its tag.
Index level	at this level, we have indices for all the word forms in the text together with their tags. The text can be viewed and queried with or without the tags. The pos-

	sibility of reverting to the raw corpus if necessary, is considered very important (Leech, forthcoming).
Lemma Index Level	this level is activated when working on lemmatized texts. It contains all the lemmas for the word forms in the text and explicit links to the forms, and to the related occurrences. The lemmas are only stored in the system in the lemma index and not explicitly at the text archive level.

The system operates automatically activating the "word path" or the "word+tag path", according to the data or operational mode selected by the user.

2.4. Query system

The query system is based on a tool box of corpus access functions; software modules from the tool box can be integrated in procedures and applications without any need to enter into the core of the function. The engine of the tool box is the query system which allows the whole corpus to be interrogated on-line, with specialized search functions for each level of query: 1) Base Functions, 2) Tag Functions, 3) Lemma Functions.

2.4.1. Base functions

These functions operate at the unannotated text level:
Search: the following types of look-up can be made:

- simple search functions can be entered using strings of characters or wildcards. It is possible to search: i) words beginning, ending with or containing a given string; ii) words beginning, ending with, containing a given string or another given string.
- complex search functions can be used to define co-occurrences of two or more given items in the same context: such functions are known as 'word families' in the system: families are defined using 'w' for item and 'f' for family with the operators AND=&, OR=/, NOT=¬. The user can retrieve contexts with: i) co-occurrences of word1 AND word2, word1 OR word2, word1 but NOT word2, ii) word3 with either word1 OR word2, iii) co-occurrences of family1 together with word4 and word5, and so on.

View: contexts which are obtained as results of a query can be displayed sorted in

(i) text order
(ii) left order
(iii) right order according to the keyword.

The size of context can be modified by the user, as he wishes.

2.4.2. Tag functions

These functions switch on the word+tag path: the system operates on the pair word+tag, considered as a simple element. The same search and view facilities as described above for the base functions are available.

In addition, searches can be made which distinguish between homographic forms on the basis of their tag. The set of tags on which the query is to operate can be defined interactively by the user (see Fig. 2).

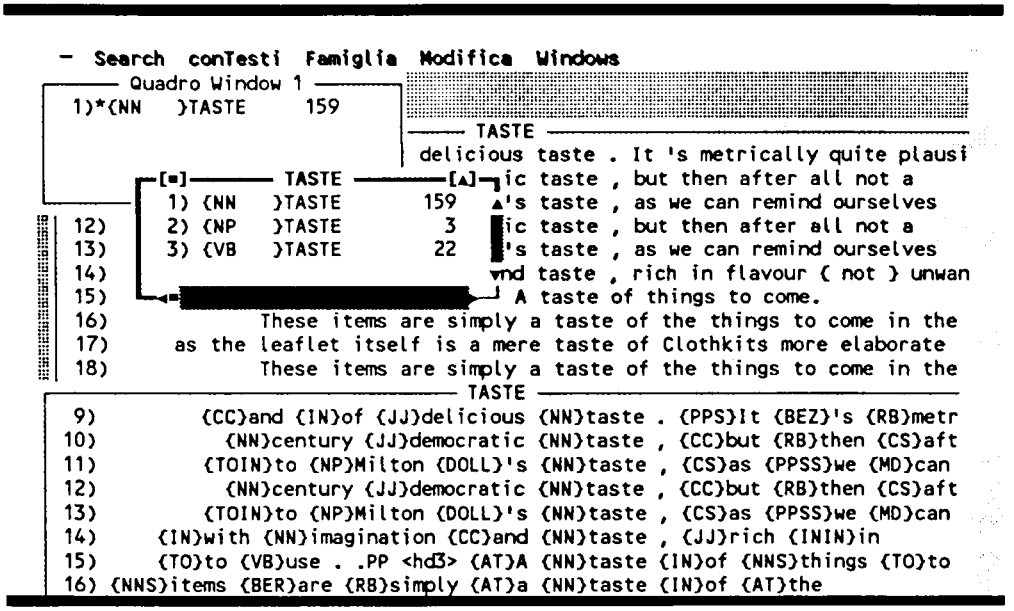


Fig. 1
Interactive selection of a word-form according to its tag.
View of contexts with or without tags

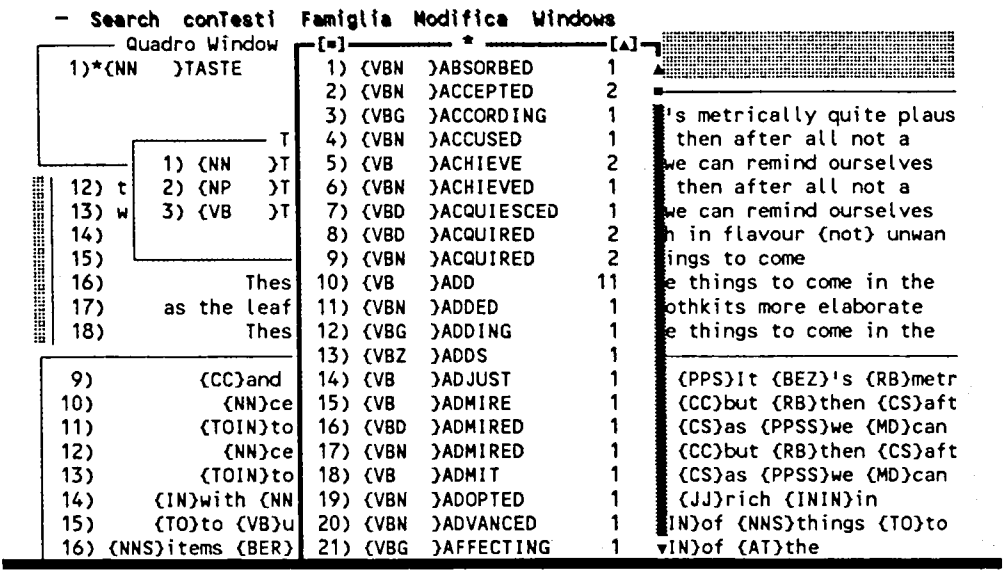


Fig. 2
Search of word-forms tagged as VERB

The context(s) retrieved as the result of a query can be displayed with or without the tags associated with each word form (see the Fig. 1).

Restrictions: in order to permit the querying of linguistic phenomena, which cannot be found by simple word+tag searches, additional functions have been implemented which permit conditions to be imposed on a query: these "restriction" rules are applied to the context(s) to be retrieved: only those contexts which satisfy the conditions imposed by the operator are accepted.

This facility makes it possible to operate selections in terms of grammatical structures, here conceived as sequences of tags. Particular sequences can be defined in order to create "macros" which can be used within the Restriction rules: the following sequence, DT [JJ*] NN [JJ*], constitutes a possible macro which could be used to search a Noun Phrase (NP).

2.4.3. Lemma functions

As we are convinced that, especially when working with highly inflected languages it can often (even if not always: see, e.g. Bindi *et al.* 1991) be useful to operate on lemmas rather than on word forms, in order to limit the dispersion of information in inflected forms.

A set of functions working at the lemma level has thus been implemented: these functions can be used to make searches on the form-lemma set, starting either from the form or from the lemma. The same types of search and view functions as described above can be used although the restriction rules can only be applied at the word-tag level, not at the lemma level, as the lemmas are only stored in the lemma index and not in the text archive.

3. Integrating the system in a Lexicographic WorkStation

In the present paper, we have described a system which has been designed to meet the needs of corpus users: it provides functions for interactive browsing and navigating through raw and/or annotated data at all the different levels of linguistic analysis.

We feel that a system of this type, used in combination with other tools implemented to meet the specific needs of the lexicographer, has an immediate application in computational lexicography and lexicology. The whole set of tools has, in fact, been used during the Tutorial in Computational Lexicography, held at EURALEX '92 (Atkins *et al.* 1992).

The query system is particularly useful during the first stages of the preparation of the dictionary entry:

— in the first scanning of corpus lines in search of relevant phenomena for a given word

— in the collection of the salient facts of a word manifested in real usage, e.g. typical left and right collocates, syntactic patterns, collocations, idioms, etc., in connection with particular statistical application procedures, such as the index of Mutual Information between word-pairs (Calzolari–Bindi 1990)

— in the acquisition of the most interesting examples, which can be picked-up directly from the corpus and put into the *polmone*, a special framework where selected contexts can be further analysed and classified by sense).

At this point, a complete Lexicographic WorkStation is available for the lexicographer. The "core" of the LWS is the lexical entry *editor*, which can

interact with the text query system (with all its access and search functions) and with the *polmone* (with the selected contexts tagged by sense) in order to work on the entry and update it with the automatic insertion of examples.

The lexicographer, thus, has simultaneously three major work components on the PC screen, integrated in a single environment:

1. all the contexts of the word under analysis (accessed by the query system)
2. the selected corpus lines which can be further analysed by sense, refined and considered as candidates for insertion in the entry (extracted using the query-system, inserted and then tagged into the *polmone*)
3. the entry, which grows step by step, as it is modified, updated, restructured, etc. (managed by the editor).

A system of this type should also be particularly useful in the storage, acquisition and restructuring of text data, for example syntactic and semantic analyses.

References

Atkins, B.T.-Calzolari, N.-Picchi, E.-Monachini, M. 1992. EURALEX '92—Tutorial on Computational Lexicography—Handout, 4 August 1992, Tampere, Finland.

Bindi, R.-Calzolari, N.-Monachini, M.-Pirrelli, V. 1991. Lexical Knowledge Acquisition from Textual Corpora: A multivariate statistic approach as an integration to traditional methodologies. In: Using Corpora. Proceedings of the Seventh Annual Conference of the UW Centre for the NEW OED and Text Research, 170–96. Oxford University Press, Oxford.

Bindi, R.-Calzolari, N.-Monachini, M.-Pirrelli V.-Zampolli, A. (forthcoming). Corpora and computational Lexica: Integration of different methodologies of lexical knowledge acquisition. Paper presented at the Corpus Workshop. Pisa, 24-26 January 1992, to be published in the Proceedings of the Conference.

Calzolari, N.-Bindi, R. 1990. Acquisition of lexical information from a large textual Italian corpus. In: Kalgren, H. (ed.): COLING '90 Proceedings, vol. 3: 54–59. Helsinki.

Calzolari, N.-Zampolli, A. 1990. Lexical databases and textual corpora a trend of convergence between computational linguistics and literary and linguistic computing. In: Hockey, S.-Ide, N. (eds): Proceedings of ALLC-ACH Conference. Oxford University Press, Oxford.

Garside, R.-Leech, G.-Sampson, G. 1987. The Computational Analysis of English—a Corpus Based Approach. Longman, London.

Leech, G. (forthcoming). Corpus annotation schemes. Paper presented at the the Corpus Workshop. Pisa, 24-26 January 1992, to be published in the Proceedings of the Conference.

Marcus, M.–Santorini, B. (forthcoming). Building very large natural language corpora: the Penn Treebank. Paper presented at the Corpus Workshop. Pisa, 24–26 January 1992, to be published in the Proceedings of the Conference.

Monachini, M. 1992. Italian Reference Corpus—Actual and Potential User Needs. NERC Working Paper, Pisa.

NERC Consortium 1991. Network for European Reference Corpora—NERC—Technical Annex of the Project. Pisa.

NERC Consortium 1992. Policy for Corpus Provision for Europe—Strategic Briefing Paper, presented at the Workshop on Language Technology, May 1992, Luxembourg.

Picchi, E. 1983. Textual database. In: Proceedings of the International Conference on Data Bases in the Humanities and Social Sciences. Rutgers University Library, New Brunswick, NJ.

Picchi, E. 1991. DBT: A textual data base system. In: Computational Lexicology and Lexicography. Special issue dedicated to Bernard Quemada. In: Cignoni, L.–Peters, C. (eds): Linguistica Computazionale VII. Pisa.

Sinclair, J.M. 1991. The automatic analysis of corpora, for the Nobel Symposium on Corpus Linguistics. Birmingham.

Sinclair, J.M. (forthcoming). Lexicographer's needs. Paper presented at the Corpus Workshop. Pisa, 24–26 January 1992 and to be published in the Proceedings of the Conference.

Zampolli, A. 1990. Project Definition for the Constitution of a Network of European Textual Reference Corpora. Pisa.

Address of the authors: Monica Monachini and Eugenio Picchi
Istituto di Linguistica Computazionale, C.N.R.
Via della Faggiola 32
56100 Pisa
Italy

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 235-263 (1992-93)

AN ANALOGICAL WAY TO LANGUAGE MODELLING: MORPHEUS*

VITO PIRRELLI-STEFANO FEDERICI

1. Introduction

The present article intends to offer an overview of a general computational approach to natural language modelling which heavily draws on the notion of analogy. In our view, a grammar, the internalized model of the linguistic intuitions of a fluent speaker, is a set of analogical relations holding among full surface forms, rather than a set of rules specifying the construction of complex forms from simple components (as within the standard generative tradition: see Anderson 1992 on this point). As emphasized by Stephen Anderson (1992, 369) this move reverses the relation between founded and founding units in language theorizing in at least one important respect: the primitive simple components making up complex linguistic units are presupposed by the latter. To give a concrete example, following Kuryłowicz (1949), the stem of a paradigm is founded on the set of fully inflected forms, rather than *vice versa*. Analogy presupposes the existence of at least two complex linguistic units—say *a* and *b*—which share a common surface "core", and a similar meaning, either at the grammatical level, or at the lexical level, or at them both.

This means that the existence of an "analogical pattern" *P* (to be defined semi-formally later in this work) is founded on *a* and *b* already existing. This assumption radically departs from the view that, e.g., *a* is basic and *b* derived from it by a rule, or that a rule originates both *a* and *b* by starting with an "abstract" *c* and adding something.

This change of perspective can be fully appreciated in the specific context of machine language learning: to learn a language means to extract a set of

* All ideas illustrated in this paper are the outcome of a cooperative effort: nevertheless, for the specific concerns of the Italian Academy, Vito Pirrelli is responsible for sections **2** and **3**, Stefano Federici for sections **1**, **4** and **5**.

Akadémiai Kiadó, Budapest

analogies from already acquired linguistic forms, for them to be extended to yet unknown forms.

Classical approaches to linguistic analogy have used the notion of analogy to explain linguistic change, that is the diachronic process by which new linguistic forms are created because they are similar to already existing ones (as in Saussure's and Bloomfield's, see Anderson 1992, 367). Accordingly, a valid analogy is always based on a general (pre-existing) rule. In the present paper we rather take a different view, namely that a general rule is always grounded on an existing analogy. This is a reasonable assumption if one wants to explore the gradual process of rule elicitation from examples. In that, our approach is reminiscent of Skousen's use of analogy, although the "analogical engine" we illustrate here is totally different from his (Skousen 1989).

In the end, there are reasons to doubt whether the distinction between a diachronic and a synchronic approach to analogy, useful though it may be in practice, is conceptually sound (Matthews 1992, 80). This work can help to shed some light on the pursuit of a unifying perspective.

1.1. Analogical lemmatisation

By lemmatisation we mean the process of associating a set of morphosyntactic features (part-of-speech, gender, number, case etc.) to word-forms, irrespective of their actual syntactic function in a given context. In lemmatisation, homographic word-forms receive more than one morphosyntactic analysis:

che relative pronoun	(English "who/which")
che declarative conjunction	(English "that")

Traditionally, pieces of software designed to handle lists of stems and inflectional paradigmatic tables of some kind, have proved to fare reasonably well in carrying out lemmatisation in this specific sense. Given a certain input word-form, they usually check off a list of available FORM/MORPHOLOGICAL ANALYSIS pairs. Such a list is either precompiled, or generated at running time. If the input form is found in the list, the corresponding morphological analysis is given; if it is not listed, the analysis fails.

Automatic lemmatisation can be made slightly more exciting by the requirements imposed on the task by the use of unrestricted, textual corpora as a test-bed. Basically, the task remains the same, but the need for coping with a virtually open-ended linguistic material makes it rather more challenging.

The general point is made that the need for coping with open-ended linguistic material calls for open-ended automatic tools of analysis to be designed.

In particular, NLP tools must be flexible enough to be able to automatically revise their own internal analysis procedures, and adjust them to new, unforeseen and some times even unexpected data. This capability will be referred to in the following as *self-modelling*. Experience in the field has shown that unrestricted data are often out of reach of the linguist's intuition, and therefore well beyond the predictive power of any rule-driven grammar whose nature and manner of operation are hand-crafted once and for all.

The line of research we sketch here draws on the notion of automatic, unsupervised learning through examples as a viable "bootstrapping" strategy for NLP. In the following we will mainly concentrate on the bootstrapping phase, as carried out by MORPHEUS, a parallel processing morphological system.

1.2. MORPHEUS: desiderata

For a self-modelling, parallel-processing lemmatiser to be a practical tool in an NLP system, we contend that a number of requirements should be met:

1) its convergence algorithm is quick enough for an effective bootstrapping strategy to be pursued, and rapid tuning onto the specific requirements of the text type in question to be attainable;
2) any training required should be fast, enabling rapid turnaround with new morphosyntactic tags and new text genres;
3) it must be computationally efficient—performing in time and space not greater than polynomial in the number of word-forms tagged;
4) it must be robust enough in its self-modelling capacity, for mistaken inductive steps to be quickly recovered and easily amended;
5) it must be flexible enough to adapt to virtually any kind of input (even ungrammatical one) with no dramatic performance degradation;
6) its overall architecture must be grounded on some fairly general, linguistically sound insights, which however do not address the task at hand in an *ad-hoc* way.

In the following we illustrate to what extent MORPHEUS meets all such *desiderata*. Before going into that, we will take a bird's eye view of parallel processing strategies and their virtues.

1.3. Parallel processing: a bird's eye view

Our learning system drew inspiration from neural network models as they have been implemented through parallel distributed processing strategies of different sorts (hereafter PDP) over the last few years (Rumelhart *et al.* 1986;

Nakamura *et al.* 1990 among others). However, the approach we have followed here departs sufficiently radically from what has been done in the field up until now. We claim that the relation between symbolic and sub-symbolic processing of linguistic data is much more complex than usually acknowledged in PDP circles (for a criticism of PDP strategies on this point, see Fodor–Pylyshyn 1988; Fodor–McLaughlin 1990). Nonetheless we endorse what we take to be the main thrust of PDP approaches, namely the parallelism of inferential operations, and the dynamic structuring of acquired knowledge, in a self-monitoring and self-adjusting network (Federici 1991).

Classical, rule-driven programs for NLP generally consist of a set of formal rules (logically, if-then implications), sequentially interpreted and applied to input items. Rules of this sort are defined once for all and stored separately from input items. The operation of each rule proceeds independently of whatever other rules may exist.

Parallel models assume that information processing takes place through the interaction of a large number of simple processing units interconnected in large networks. Units can be conceived of as high-level equivalents of neurons in the brain, but the analogy is merely suggestive and arguably slightly misleading too (Smolensky 1988). Each processing unit notionally represents a prime at the level of analysis the network is expected to perform.[1]

The most important entities involved in PDP-processing are complex patterns of activity over many units. Each pattern consists of the simultaneous activation of mutually supporting units, and corresponds to a certain (partial) hypothesis/response of the system to the input/stimulus processed (e.g., the identity of a certain string of characters). Each unit may participate in many such patterns (a certain character may occur in a certain position in many words). Therefore, more hypotheses can easily be conjured up and contemplated at the same time, through their being partially "turned on" by the same set of input units. This generates an unstable state of the system's network, which has to "cool off" for the system to reach a stable state and converge onto one (or more) response(s) to the input at stake.

[1] For example, given the task of recognizing strings of characters as words, and given a certain input string of characters we want to identify as a word-form of a certain type, in a PDP-system the hypothesis about the identity of this word is distributed over a large number of primitive units being simultaneously activated. Such units are not words themselves (as it would be the case in a classical list-based approach), but rather characters, or, more appropriately if one wants the system to be able to 'read' noisy input, sub-characters: e.g., vertical, horizontal or slanting lines making up block letters in written input (Rumelhart *et al.* 1992).

Extracted patterns of regularities (as opposed to rules) have no implications singly. It is only the entire set of regularities that has any implications. Inference must be a cooperative process. The crucial role of such a cooperation comes out very clearly in a very simple case of the word-recognition task. Suppose we are faced with a string of characters having one character missing (say M I S S ? N G), and suppose we want the system to guess which word it stands for. The problem a sequentially operating logical system is faced with is that, in order to provide the missing letter, the system has to know first which word it is dealing with; however for it to pick up the right word in its list, it has to know first which character is missing. This is clearly a paradoxical state of affairs. The simultaneous activation of multiple, partially activated solutions in a parallel network is capable of getting it around.

The building up of such a network of interconnected units is carried out through training: the system has to learn how to associate a certain response to a certain input, by being exposed to a number of correct stimulus/response pairs. Certainly, given the unsupervised nature of the learning process, the network can be skewed, and the system can be led astray. However, it is contended that, by adding further regularities, conclusions that were formerly valid but that proved wrong at a later stage can be repealed: in this respect, "parallel inference is fundamentally non-monotonic" (Smolensky 1988).

Last but not least, no principled line is drawn to separate rules from the items to which they apply. In fact, there is no question whether a given pattern of units should be stored directly in a repertoire of idiosyncracies (the Lexicon) or should rather be stocked in a set of more general statements about admissible input (a grammar). General patterns are elicited through a process of on-line generalization over particular patterns. More precisely, the process of generalization (regularity extraction) is the natural by-product of the storage-retrieval machinery of the system itself, which exploits the overall rate of associations/similarities among already learned patterns. In a sense, the network of interconnected units is a huge Lexicon itself which produces its own set of general statements by establishing excitatory/inhibitory links among partially idiosyncratic patterns (a similar model for a morphological Lexicon is suggested by Bybee 1988).

2. Analogical modelling of language

In this section we will outline a set of straightforward principles which are intended to model an analogical view of language. Following Skousen (1989),

a list of abstract cases can be defined as follows, and a categorical classification of each item in the list is provided:

i)	127	*x*	028	*o*
	137	*x*	338	*o*
	037	*x*	348	*o*
	337	*x*	038	*o*
	227	*x*		

Each item in the left-hand column represents a concrete instantiation of a linguistic context: e.g., a word-form, a phrase, a sentence etc. Such instantiations are *exempla*. Arabic numerals are used to define three position-dependent variables, whose values range as follows:

VARIABLE	VALUES
1	0,1,2,3
2	2,3,4,5
3	6,7,8,9

where "variable 1" means 'variable in first position in the string', "variable 2" 'variable in second position' and so on and so forth. We use numerals as values as well, to specify the primitive units out of which *exempla* are made up. We take a dash ("–") to mean: 'any value in the range of admissible values for variables 1, 2 or 3'. On the right-hand column of i), the appropriate category of *exempla* is given. Each category is expressed by a letter (*x*, *o*), and assumed, for the sake of simplicity, to be atomic.

Generally speaking, the rule-driven way of describing the categorical behaviour of the list of *exempla* in i) is to write the simplest possible set of rules, provided they correctly describe such a behaviour:

ii) $-7 \Rightarrow x$
$-8 \Rightarrow o$

We refer to ii) as to the "rule-driven way" to generalization. It contrasts with other conceivable ways to express the same facts in i), since it does not unnecessarily increase the number of symbols needed to describe the linguistic behaviour correctly (formal parsimony).

The generalization in ii) holds on the assumption that i) is representative of the entire population we aim at describing. Otherwise, the description in ii)

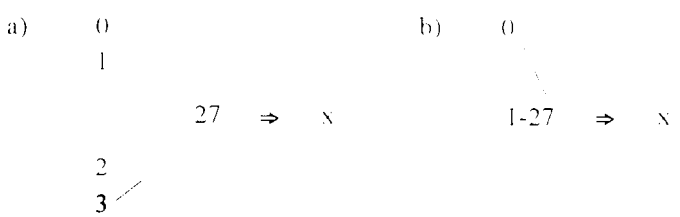
would be inaccurate. What ii) says is that any item ending in "7" is of category x, unless otherwise specified. On the other hand, the type of evidence provided by i) suggests that any item ending either in "*27*" or "*37*" is associated to category x:

iii) $-27 \Rightarrow x$
$-37 \Rightarrow x$

In modelling learning practices in any normative context, the underdeterminedness of the evidence elicited from *exempla* is a fact. We resolved to squarely face it. The model of analogy-based inference which is implemented in MORPHEUS would converge on to a iii)-like statement rather than a ii)-like one, namely:

iv) "27 is classified as x when preceded by {0,1,2,3}"

where {0,1,2,3} defines the set of all available alternatives encountered so far. The generalization in iv) is stated in contextual terms. "27" is called the **core** of iv). The set which keeps company with it, is its **contextual specification**. The *core* is what is shared by more *exempla* classified the same way: e.g. having the same x category. Generally speaking, given the same core, the bigger the contextual specification (the more alternatives it contains), the more powerful the conveyed grammatical statement. The following diagram can help in giving a pictorial justification of the just introduced notions:



a) and b) are called **Analogical Patterns**. We have implemented such analogical patterns as clusters of units in a parallel-network. Accordingly, our system does not put forward rules in a strict sense. It only structures its own experience (the learned *exempla*), and interprets new, unknown evidence, in the light of that experience, by analogy to what is known. Whenever the network is exposed to a new string, all activatable patterns will be turned on, and give their

response (their OUTPUT). Eventually, the system will settle down onto the most plausible pattern (response), which will then be yielded as the winning output. Intuitively, the "winner" is that pattern whose core most extensively "overlaps" with the input string. From a formal point of view, pattern activation hinges on two auxiliary notions, namely "activation ratio" and "analogy ratio" which can be defined as follows:

$$\textit{activation ratio} = \frac{\textit{number of numerals of the activated core contained in input}}{\textit{total number of numerals in the activated core}}$$

$$\textit{analogy ratio} = \frac{\textit{total number of numerals in the core}}{\textit{total number of numerals in input}}$$

Their interaction is described in detail in Pirrelli (1993). Intuitively, the pattern which is yielded by MORPHEUS as a response is the one with the highest activation ratio.

This set of straightforward principles can be extended in such a way that an entire range of significant properties, defined at different levels of linguistic analysis, can be made to interplay in a highly interactive, parallel fashion. If strings are surface lexical entries, and categories the complex of phonological, morphological, syntactic and semantic features by which entries are defined, a network of analogical cores can capture a good many levels of lexical redundancy. It can thus be regarded as a dynamic, self-modelling lexicon which infers a significant amount of linguistic generalizations and can eventually be used as a parser (Pirrelli 1993).

3. MORPHEUS: the architecture

The learning system we have in mind is an *interactive*, *multifunctional* tool. We can factor out the system's learning steps as follows:

1) an INPUT ("raw text") is given by a hypothetical "teacher"; MORPHEUS conjures up one or more tentative response(s) on the basis of both already known *exempla* and extracted cores,

2) a (hypothetical) teacher provides the expected response in context (lemmatised text),
3) the system revises its own data network in the light of 2), if need be.

By dropping steps 2) and 3), the system acts as an ordinary parser. When the learning "switch" only is on, MORPHEUS stops outputting its own response(s). A reasonable assumption is that the system works as a parser when its learning phase has been brought to completion. The interplay of steps 1), 2) and 3) is pictured in Table 2 in the Appendix. In that chart, three data structures (CORES, EXEMPLA and BUFFER) and two operations, namely **string matching** and **pruning**, are illustrated. Something has already been said about the structure of EXEMPLA and CORES (analogical patterns and their parallel architecture) in the previous section: more will be said in sections **3.2** and **3.3**. As to the role of BUFFER, the discussion is delayed until "pruning" is dealt with (sections **3.4** and **4.1**). The following section is devoted to a formal discussion of "string-matching".

3.1. How do we define string-matching?

So far we have been tacitly assuming some form of string-matching being at work in the process of:

1) activation of analogical patterns by inputting new, unknown strings;
2) comparison between new *exempla* and acquired data;
3) extraction of shared (sub)strings having the same category.

String-matching is therefore at the core of analogical parsing (as symbolically represented in Table 2 in the Appendix), a pervasive operation affecting the learning routine as well as the parsing strategy.

In principle, string-matching can be defined in a number of different ways. In the following we will review only three of them, to give the reader a flavour of the range of alternatives.

First, the following formal definition is introduced:

> "a *string-matches* b on *s* if there is a common substring *s* which both of them contain"

where a and b are numeric strings ("147", "308" etc.) of the type we have been using so far.

s can be constrained in a number of ways. We will show three alternative such ways, starting from the loosest one to end up with the most constrained one.

We will make use of variable "x", ranging over single numerals in the string. We define the function "length (*s*)". It takes a string as argument, to

yield a numeric value, namely the length of the string: for example, length (63008) = 5. The "concatenation operator" is indicated by "," (comma). For example, if "s = 327", then "s,1 = 3271". All alternatives are defined as follows:

1) string-matching by "**linear patterning**" is recursively defined on length (s):

   a) length (s) = 1; a *string-matches* b on s, if
   "a = a_1,s,a_2" "b = b_1,s,b_2" for a_i and $b_i \in N^*$;

   b) length (s) = n+1; a *string-matches* b on s, if
   there exist $s_0 \in N^+$, $x \in N$: length(s_0) = n, "s = s_0,x" and
   "a = a_1,x,a_2" "b = b_1,x,b_2" and a_1 string-matches b_1 on s_0, for a_1 and $b_1 \in N^+$, and a_2 and $b_2 \in N^*$.

   Example:

   according to string-matching by "linear patterning"

   1437 **string-matches** *1307* **on** *137*
   1763 **does not string-match** *1307* **on** *137*

2) string-matching by "**linear sequencing**"

   a *string-matches* b on s by linear sequencing, if

   "a = a_1,s,a_2 "b = b_1,s,b_2" for a_i and $b_i \in N^*$, $s \in N^+$;

   Example:

   according to string-matching by "linear sequencing":

   385084 **string-matches** *50822* **on** *508*
   5408 **does not string-match** *5083* **on** *508*

3) string-matching by "**strict overlapping**"

   a *string-matches* b on s by strict overlapping, if

   "a = a_1,s,a_2" "b = b_1,s,b_2" for a_i and
   $b_i \in N^*$, $s \in N^+$; and length(a_1) = length(b_1);

   Example:

   according to string-matching by "strict overlapping"

47300 **string-matches** *47358* **on** *473*
20473 **does not string-match** *47350* **on** *473*

To sum up, string-matching can be more or less constrained, thus yielding different results. Other types of string-matching can be conceived of: in particular, in the present work we will be concerned with a different sort of string-matching by overlapping, namely what we call "string-matching by *tail overlapping*". We can define it thus:

4) string-matching by "**tail overlapping**"

a *string-matches* b on *s* by tail overlapping, if

"a = a_1,*s*" "b = b_1,*s*" for a_1 and b_1 N^* and $s \in N^+$;

Example:

according to string-matching by "tail overlapping"

00473 **string-matches** *5473* **on** *473*
20473 **does not string-match** *47350* **on** *473*

3.2. An illustrative example

So far, we have deliberately oversimplified the matter to a certain extent. In section **2** we have been assuming that each categorical output is an atomic feature (a letter). Clearly, this is not true of our application, where the output is a string of morphosyntactic features:

credeva ⇒ **credere/v 3/v s/v imp/v ind/v**

In this section we will take a glimpse at the dynamic functioning of the system through a simple series of consecutive learning sessions, to show rather cursorily how the principles outlined in section **2** naturally apply in MORPHEUS.

A straightforward way to represent the pattern of association between input and output strings is given in Fig. 1, Table 1 of the Appendix. The input and output levels are associated through activation links, represented in Table 1 as solid lines. Core extraction can apply to both levels. Extracted cores are represented as separate patterns, as shown in Fig. 2, which illustrates the internal representation of the patterns below, after they have been learned:

PATTERN 1 = (*credeva* ⇒ **credere/v 3/v s/v imp/v ind/v**)
PATTERN 2 = (*temeva* ⇒ **credere/v 3/v s/v imp/v ind/v**)

The reader should note that now the string *credeva* is split into two parts (two input cores), namely *cred* and *eva*, which are linked to two output cores, namely the lemma *credere*, and the morphosyntactic feature string "3 sing imp ind" (third person singular, imperfect indicative). We refer to such a split-up representation as a "compositional representation". Incidentally, the string *credeva* is now assigned two different representations, since the original pattern (Fig. 1) is not replaced by the compositional representation of Fig. 2: the two representations are simply simultaneously built up in the network. As will be shown later in this work, the system is well capable of coping with the existence of multiple patterns with the same morphological features with no problems of multiple equivalent responses. As to the meaning and function of the solid arcs dangling from the output nodes of each pattern more will be said in the following section.

It may be worth noting in passing that the coexistence, in the network, of whole-word-form representations as in Fig. 1 (whenever available, that is, for known tokens) and of compositional representations such as in Fig. 2 (induced by the system itself through core extraction) has an interesting psycholinguistic plausibility (Caramazza *et al.* 1985, 1988).

Let us suppose now that a further new pattern is fed into the system:

PATTERN 3 = (*credemmo* ⇒ **credere/v 1/v p/v perf/v ind/v**)

Again, a simple application of analogical principles will yield the overall network configuration illustrated in Fig. 3. Note that the system has now stripped off one more inflectional ending (one more core).

Let us consider now the system's performance in the analysis phase. Three strings are evaluated which the system has never encountered before: namely *tenemmo*, *volemmo* and *voleva* in this order. The system's responses will be as follows:

INPUT1 = *tememmo* ⇒ OUTPUT1 = **temere/v 1/v p/v perf/v ind/v**
INPUT2 = *volemmo* ⇒ OUTPUT2 = **?/v 1/v p/v perf/v ind/v**
INPUT3 = *voleva* ⇒ OUTPUT3 = **volere/v 3/v s/v imp/v ind/v**

INPUT1 illustrates a case where paradigm extension (see below) has made it possible to reconstruct the entire surface form compositionally. The question

mark in OUTPUT2 stands for missing information: the system does not know of any lemma for *volemmo* at this stage yet. If question mark is replaced by the appropriate lemma (taken from the training corpus) MORPHEUS is able to produce the entire morphological analysis of a string such as *voleva*, including the lemma, as illustrated by the INPUT3 case above.

Clearly, erroneous guesses are always possible, since generalization through induction cannot be constrained *a priori*. Our experience, however, shows that the system's built-in error-shooting and error-amending capabilities are smart enough for the accuracy-rate of its performance to grow at a remarkably high rate (see performance-figures, Tables 5 and 6 in the Appendix), comparatively much faster than by using ordinary stochastic approaches (e.g., Church 1988; De Rose 1988). A thorough description of such built-in capabilities would go well beyond the scope of this article. More on this can be found in Pirrelli (1993). Suffice it to say at this juncture, that cores, as illustrated in section 2 above, are built up monotonically. Basically, this means that no backtracking of previously hypothesized generalizations is ever needed. The only two possible ways in which the system can be duped into making a mistaken inference, namely so-called "background noise" (when too specific properties form a core) and so-called "overshooting" (when too general properties have been accidentally extracted) are due to an unfortunate sequence of data being stumbled upon. Hence they can be easily got around through exposure to further more appropriate evidence, which is bound to carry more analogical weight and override either too *ad hoc* patterns or too general ones.

3.3. Paradigm extension and exceptions

Let us turn back to Fig. 3 in Table 1. The solid arcs which hang from the bottom of each pattern represent paradigmatic links (referred to in the following also as "inter-unit links"). In MORPHEUS, paradigms are set up through linking inflectional endings with the lemma from which they have been stripped off. Figure 3 in Table 1 of the Appendix represents a case of indirect paradigmatic link: the node *credere* is linked to both *-emmo* and *-eva*, which are, as a result, related to each other.

Paradigm extension is modelled through extending constellations of inflectional endings to other lemmata. This is based on the simple idea that once a given word takes a particular inflectional ending, it has to take all other paradigmatically related inflectional endings too (see how *temere* gets linked to *-emmo* in Table 1, Fig. 3); otherwise it is an exception.

Exceptions are taken to be idiosyncratic patterns which are never broken down into smaller cores. This means that they are "whole-cored" patterns.

Hence they cannot possibly fit already established patterns of regularities. Note that exceptions fail to behave compositionally; this implies that it is not possible for the system to further decompose the corresponding OUTPUT node into simpler nodes. A typical exception thus looks like the pattern of *credeva* in Table 1, Fig. 1, in the Appendix. Exceptions always override more regular patterns. This is not stipulated in any *ad hoc* way, but descends from general analogical principles. A fully activated whole-cored pattern always overrides fully activated smaller cores paradigmatically linked to one another.

3.4. Learning through examples: a realistic solution to blind-search algorithms

In this section we want to make the point that any form of more or less unconstrained string-matching (e.g., "linear patterning" in section **3.1**) does not raise problems of either combinatorial explosion or hard computational tractability in analogical parsing, since the space of logical search available to the learning system progressively narrows down as it accumulates more and more evidence. In fact, the more the system knows, the less blind is its string matching routine, and the less effort it will take. Hence, learning does not affect only the range of data to be memorized, but also the **way** relevant cores are looked for. In one word, learning concerns not only what to search, but also how to do it.

First of all, the reader should recall that a core is extracted by the system on condition that a common output (a linguistically relevant interpretation) is shared by the INPUT string and some already existing string in the data network (*exempla*). Hence, string-matching will never result in brute-force, blind search.

Secondly, there is a further important constraint at work, according to which not all possible hypotheses, out of the set of all candidate "cores" extracted through unconstrained string-matching and temporarily stored in BUFFER, are retained by the system for permanent storage in CORES (see Table 2 in the Appendix). For each new word-form being learned, only the "strongest" hypothesis, i.e. the longest extracted core, is actually stored in CORES. The process of picking out one core only at a time, indicated as "pruning" in Table 2, significantly curtails both time and space taken by the system in building up CORES.

Pruning has another important side-effect: it makes the system biased towards rather specific cores. Such a bias, as illustrated in what follows, improves on the linguistic meaningfulness of cores themselves. In fact, at early stages of the network-building phase, all "winning" cores are likely to be comparatively short (see rank-values in Table 3a in the Appendix, on which more in section

3.5). So the system is likely to have more than one "winning" candidate in BUFFER. In this case, the system's practical choice is to store into CORES one core only from BUFFER. This is a random choice: as a result, chances are that the extracted core is just a linguistic nonsense. However, candidate cores in BUFFER will become longer and longer as the system accumulates a fair amount of learned evidence. By pruning out shorter cores, the system is bound to retain more and more linguistically significant cores, for reasons illustrated in the following section.

In case of more than one winning core in BUFFER, a better strategy than random selection would be choosing that core which already exists in CORES and has got more paradigmatic links. This sort of interplay between CORES and PRUNING (mediated through paradigms) has interesting consequences which are currently being explored. Mainly, it interferes with STRING-MATCHING in such a way that the latter adapts to the morphological structure of the language of the training corpus. For example, in a language like Italian, word endings will be extracted more easily, since they readily extend to a greater number of words: hence, unconstrained string-matching would, in practice, be equivalent to "tail overlapping".

3.5. Pseudo-affixes and their linguistic meaning

What sort of linguistic meaning is carried by extracted cores?

Table 3b in the Appendix illustrates the analogical pattern associated to the morpheme INFINITIVE in MORPHEUS's network after training. The two lists on the same page (Table 3a) contain substrings which have been given the label INFINITIVE by training MORPHEUS on COMPLEX and NEWS. We refer to these substrings as "pseudo-affixes" (an analogous use in Wothke 1986). Their linguistic interpretation might appear rather moot. In fact, they have a fairly straightforward meaning. What pseudo-affixes in Table 3b have in common (namely the substring *-re*, highlighted in the graph) is akin to the notion of morpheme as 'minimal linguistic sign'. Pseudo-affixes act as 'contextual disambiguators' of the morpheme they are linked to. To put it rather formally, given a morpheme B whose morph is b, a pseudo-affix ab, linked to B, says that the string b is to be interpreted as B when preceded by a (see below for a concrete example). This statement mirrors the analogical standpoint epitomized in iv) in section **2** above. Hence, in MORPHEUS, *-re* as such plays a rather marginal role, and is encoded accordingly. This is shown by the list in 9a), where each substring is accompanied by two values: its rank (which indicates the order of extraction), and its coefficient of use ("c.u.") after training, which indicates how many times that particular input core has been

the biggest one (and thus the "winner") in giving an appropriate system's response. For example, *-re* has been extracted at an early stage, but it has never been a "winner" after training.

This shows how some well-established linguistic notions such as "morpheme" turn out to have a rather loose counterpart in analogical terms. This does not have to worry us. It can be argued that not all the formal notions linguists have come up with lend themselves to a straightforward procedural use, as principles governing the way language is learned, analysed and produced. Linguistic concepts are the outcome of an accurate process of generalization over data, which describes them through the most possible parsimonious set of primitives. This is a legitimate step, as long as one does not automatically assume that the internalized grammar of a fluent speaker, has been forged through the very same process, to end up manipulating the very same set of primitives linguists argue for.

Incidentally, the reader should note that in spite of their being the outcome of a training phase carried out on substantially different texts, the two lists are remarkably similar as to their form and rank. This means that random factors concerning data distribution in the training phase have a rather negligible incidence on pseudo-affix extraction, and on the role they play in the analysis phase.

Not every pseudo-affix stored in CORES will remain there for ever. Many of them are actually discarded after a while, on the reasonable assumption that, in machine learning, under-utilized patterns are simply "forgotten". "Forgetfulness" can in theory be implemented in two ways:

1) by eliminating those patterns which after testing have a zero coefficient of use;

2) by discarding only those patterns which, after having been activated over testing, have never prevailed over other rival patterns (that is, they have never been winners).

We are persuaded that 2) is a better strategy than 1), although, in many cases, they both yield the same practical result. Table 3c illustrates the role of pseudo-affixes in MORPHEUS's response. Suppose the system has extracted the sequence *-eva* twice, linked to two different morphemes: namely 'imperfect, 3 person singular' from, say, *credeva* (English '(s)he believed') and *rompeva* (English '(s)he crashed'), and 'present, 3 person singular' from, say, *solleva* (English '(s)he lifts') and *inneva* (English 'it snows down'). Since the same string supports both hypotheses, MORPHEUS is in no position to make the right choice when faced with a word-form ending in *-eva*. It then will give both responses. Suppose now, that MORPHEUS finds that the string *-deva*

is linked to 'imperfect, 3 person singular' only. Then, when confronted with, say, *rideva*, MORPHEUS can now yield a single response, namely 'imperfect, 3 person singular', since this hypothesis is supported by a longer activated string.

4. Testing MORPHEUS

In Federici–Pirrelli (1992), MORPHEUS was trained on a excerpt of 4,000 word-forms (henceforth COMPLEX), drawn from a fairy-tale by Italo Calvino. It was then tested as a lemmatiser on the following 4,000 word-forms of the same text. Results were extremely encouraging, with an accuracy rate averaging 92% of correct analyses. Correctness is evaluated by comparing the system's output with the manually disambiguated version of the same text. As mentioned before, the system's response is always the pattern with the highest "analogy ratio" among all activated patterns. If more than one response is given the same highest score, the system is in a stalemate: thus it yields more than one solution. For example, this is the case of a morphologically ambiguous word-form which has been assigned two or more different analyses in previous occurrences. If the system's response does not contain the analysis annotated in the text, an error is counted. Finally, MORPHEUS provides also "partial analyses", when a stem is not recognized, but a categorial classification is tentatively produced all the same. In the following, correct partial analyses are counted as correct responses too (see Table 4 in the Appendix, where correct partial analyses are referred to as "correctly categorized forms"). The reader should note that, by these criteria, due to the existence of homographs, MORPHEUS can fully recognize word-forms and nonetheless assign them a wrong analysis. Incidentally, this contributes 70% of the error rate scored by MORPHEUS.

We report here figures concerning the performance of a modified version of the system, trained on two substantially different types of text, which are tagged according to three different schemata. Further developments will eventually be delineated, and preliminary conclusions will be drawn.

4.1. Pruning out shorter overlappings

Overlappings between input sequences and analogical patterns are defined in terms of some elementary string-matching operations. In our first testing, we decided to extract all possible sequential matches (by tail-overlapping), no matter how long they were. The idea was that smaller extracted analogical patterns would have improved the inferential capacity of the system. We then

moved on to a simplified assumption: at each learning session, only one string-match is added to the network as a new analogical pattern, namely the longest core. All the remaining overlappings (if any) are discarded. As already pointed out in section **3.4**, such a form of pruning is convenient in a number of respects. For the present purposes, it is worthnoting that, by pruning out shorter cores, time and space complexity of the system becomes much closer to a function which is linear to the number of word-forms. In Tables 5 and 6 solid lines show space growth when pruning is activated, while dotted lines stand for space growth without pruning.

In Tables 7 and 8 in Appendix, the system's performances are scanned as a function of learned word-forms (the x-axis). Figures concerning known word-forms ("k.w.", i.e. word-forms already encountered by the system in the learning phase) are factored out of the overall rate of MORPHEUS's correct responses ("c.r."), to evaluate the effectiveness of its inferential routine. Note that the incidence of the inductive routine grows faster than the amount of repeatedly encountered word-forms, irrespective of the pruning/non-pruning algorithm. As expected, the "pruned" version is slightly slower in making the right inferential choices, but this affects the early stages of the learning phase only. In the end, performance degradation disappears. The basic insight is that, when you can rely on enough training data, you can wait for the appropriate example to come. The wealth of direct evidence makes up for weaker inferential capacities (the same point, but in a different context, is made in Brent 1991).

4.2. Changing text type and tag set

How flexible is analogical pattern extraction? To address this question, we carried out several tests on substantially different training texts of comparable size, and compared the results (Table 9 in Appendix). First, the same text as in COMPLEX was tagged with a different tag set (hereafter referred to as CALVINO).

A tag for a noun in COMPLEX looks like the following:

marea marea/n f/n s/n

which says that *marea* (English "tide") is a feminine (f/n), singular (s/n) noun.

Similarly, each word-form in CALVINO is analysed as a lemma plus a tag. The tag is taken as an atomic unit by MORPHEUS. However, the set of grammatical categories is far more granular than in COMPLEX, since parts-of-speech are specified at a deeper level of detail. Moreover, morphosyntactic information (e.g., gender, number etc.) is expressed through class markers

(say the category "N" is used for singular nouns, "NS" for plural nouns), not through features (as in COMPLEX, see above). This state of affairs multiplies the range of possible ambiguities for each word-form, and increases the likelihood of MORPHEUS making the wrong choice.

In a third experiment we used a sample representative of a different text-type. This differs from both COMPLEX and CALVINO in two important respects:

1) it is not homogeneous in text genre

2) it is morphosyntactically annotated in a different manner.

As to 1), the new training text is a collection of short magazine papers covering the most disparate topics (hereafter NEWS). As to 2), the tag-set is even more granular than in CALVINO, allowing for rather subtle inflectional distinctions (e.g., beside verbal person and number, tense and mode are specified as well).

Performances are shown in the Appendix, where success-rates average 94% (NEWS) and 95% (CALVINO). In Table 9, we have separately counted correctly categorized word-forms ("c.c." in the Table), and correctly lemmatised ones ("c.l."). In the former case, only the (correct) tag is yielded by MORPHEUS. In the latter, both the tag and the appropriate lemma. The sum of the two figures gives the overall accuracy rate of the system's performance. The reader should note that: i) lemmatised responses are much less frequent in NEWS than in CALVINO; ii) lack of lemmatised responses is made up for by an increase in categorized responses; iii) the fork between "c.c." rates and "c.l." rates widens as the system learns more input; iv) such a fork widens more quickly in CALVINO than in NEWS; v) the sum of c.c. rates and c.l. rates is virtually the same in the two texts for $n > 500$. We then conclude that lack of lexical homogeneity a) affects the quality of the learning routine's performance; b) morphological regularities are extracted irrespective of the text genre (a confirmation of results reported in **3.5** above); c) lexical acquisition is much quicker in homogeneous texts. In the following we will examine how differences in corpus composition relate to differences in the curve of growth of the resulting networks.

In Table 4 in the Appendix, the incidence of various elements on the overall accuracy rate of MORPHEUS is shown. The first histogram illustrates the prevalence of known words in the system's correct responses when a text sample (10000 tokens from NEWS) is given as input. Comparatively, the relevance of paradigm-extension is rather poor, as shown by the amount of correctly lemmatised unknown forms. This contrasts with the system's accuracy rates in the remaining histograms. This time two samples of verbs only (1000 and

2000 tokens from NEWS respectively) are given as input. Accordingly, the amount of correctly lemmatised word forms increases noticeably (up to 10% of the overall accuracy rate): this bears witness to the effective contribution of paradigm extension to analogical lemmatisation.

4.3. Network's complexity and corpus composition

Although pseudo-affix extraction could potentially lead to an exponential growth in time and space, this is actually not the case in our system.

The possible string-matching procedures illustrated above are in fact only subcases of the general string-matching algorithm, whose complexity can be stated thus:

$$\text{string-matching complexity} = O(2^L) \text{ with } L = \text{max word-form length}$$

such a complexity is reached when one extracts all common substrings (cores) shared by two word-forms. In this case, after N word-forms having been processed, the number of potential inter-unit links in the network (the worst-case structure in terms of allocated space) will be:

$$O(\text{(Sum(i = 1,N)(i-1)*}2^L)^2) = O(N^4 * 2^{2L})$$

where at the i-th step we can potentially extract 2^L cores by comparing each of i-1 word-forms learned at this stage (i.e., $(i-1)*2^L$ new units) with new input. This way we can create up to $((i-1)*2^L))^2$ inter-unit links.

As a matter of fact, the string-matching routine implemented in our system does not make the network grow at such an exponential speed. Extraction takes place when and only when an overlapping word-ending is found (i.e., in case of tail-overlapping, $2^L \rightarrow 1$), which keeps real growth down to the following worst case:

$$\text{potential link number} = O(\text{(Sum(i = 1,N)(i-1))}^2) = O(N^4)$$

This can be further lowered when, thanks to "buffer pruning" (the system does) nothing more than one core extraction per word-form at a time (namely the longest possible one: hence $i-1 \rightarrow 1$). We thus achieve:

$$\text{potential link number} = O(N^2)$$

At this point the following question arises: does the system actually stumble upon the worst case ever? The answer is: no, it does not. As Tables 5 and 6 in the Appendix show, the real space and time growth is significantly lower than expected: we are actually very close to a linear growth. The result is not fortuitous, as confirmed by the converging results obtained from the separate tests carried out on different samples. How does complexity relate to corpus composition? Table 10 in the Appendix shows that the growth in number of links is a function of two contextual factors: the granularity of the set of morphosyntactic labels and the overall number of encountered lemmata. Clearly, more new words (lemmata) mean more nodes in the network: hence more potential links to be set up. On the other hand, however, an increase in categorial granularity lowers down the number of extracted cores, and thus decreases the number of associations (inter-unit links) among paradigmatically related lemmata. This explains why COMPLEX has the richest network structure, CALVINO (which is the same corpus as COMPLEX but with a different, more granular tag-set) has the poorest one, while NEWS takes an intermediate position between the two. The conclusion accords well with the fact that NEWS is the least lexically homogeneous text of our three samples.

5. Conclusions

Our experience with MORPHEUS prompts the following remarks: i) fast-converging parallel networks do not necessarily need long, spoon-fed drilling; ii) machine learning does not call for massive, built-in, domain-specific knowledge: fairly general principles of analogical parsing do nicely the job; iii) the use of large text samples makes up for weak inferential capacities; iv) the use of textual excerpts paves the way to some interesting applications in text analysis; v) theoretically grounded linguistic notions (like, e.g., "morpheme") and principles (like, e.g., "formal parsimony") can play second fiddle in some NLP systems. Last but not least, the job of tuning the system's performances to the requirements of a particular text type is not added on top of MORPHEUS architecture, but is a rather natural spin-off of its self-modelling capacities.

As observed above, some components in MORPHEUS can be made interact more closely than they do now. We are currently assessing the incidence of the interaction between CORES and PRUNING on the system's performances. Along the same lines, MORPHEUS architecture can be revised in such a way that STRING MATCHING is triggered only when the system makes a mistake. By doing this way, MORPHEUS takes an inferential step only when some

negative evidence forces it to do so. Arguably, this can improve the system's accuracy in some cases.

References

Anderson, S.R. 1992. A Morphous Morphology. Cambridge University Press, Cambridge.

Brent, M.R. 1991. Automatic acquisition of subcategorization frames from untagged, free-text corpora. In: Proceedings of the 29th Meeting of the ACL.

Bybee, J.L. 1988. Morphology as lexical organization. In: Hammond, M.–Noonan, M. (eds): Theoretical Morphology. Approaches in Modern Linguistics. Academic Press, San Diego.

Caramazza, A.G.–Miceli, M.C.–Silveri, A.–Laudanna, C. 1985. Reading mechanisms and the organization of the lexicon: Evidence from acquired dyslexia. In: Cognitive Neuropsychology 2: 81–114.

Caramazza, A.–Laudanna, A.–Romani, C. 1988. Lexical access and inflectional morphology. In: Cognition 28: 297–32.

Church, K.W. 1988. Stochastic parts program and noun phrase parser for unrestricted text. In: Proceedings of the Second Conference on Applied Natural Language Processing, 136–43.

De Rose, S. 1988. Grammatical Category Disambiguation by Statistical Optimization. In: Computational Linguistics 14.

Federici, S. 1991. SECS-NLC: A Self-Expandable Connectionist System for Natural Language Comprehension. In: Proceedings of AAAI Spring Symposium.

Federici, S.–Pirrelli, V. 1992. A bootstrapping strategy for lemmatisation: learning through examples. In: Proceedings of COMPLEX. Budapest.

Fodor, J.A.–Pylishyn Z.W. 1988. Connectionism and cognitive architecture: a critical analysis. In: Cognition 28: 3–71.

Fodor, J.–McLaughlin, B.P. 1990. Connectionism and the problem of systematicity: Why Smolensky's solution doesn't work. In: Cognition 35: 183–204.

Kuryłowicz, J. 1949. La Nature des procès dits "analogiques". Acta Linguistica Hafniensa 5: 1–37.

Matthews, P.H. 1992. Morphology. Cambridge University Press, Cambridge.

Nakamura, M.–Maruyama, K.–Kawabata, T.–Shikano, K. 1990. Neural network approach to word category prediction for English texts. In: Proceedings of COLING 90.

Pirrelli, V. 1993. Morphology, analogy and machine translation. Ph.D. dissertation. Salford University (forthcoming).

Rumelhart, D.E. *et al.* 1986. Parallel Distributed Processing. MIT Press, Cambridge, MA.

Skousen, R. 1989. Analogical Modelling of Language. Kluwer, Dordrecht.

Smolensky P. 1988. On the proper treatment of connectionism. In: Behavioral and Brain Sciences 11: 1–74

Wothke, K. 1986 Machine learning of morphological rules by generalization and analogy. In: 11th COLING (ACL). Bonn.

Appendix

Table 1

Fig. 1

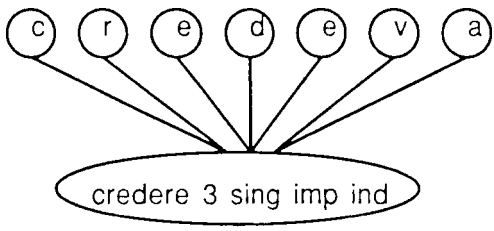


Fig. 2

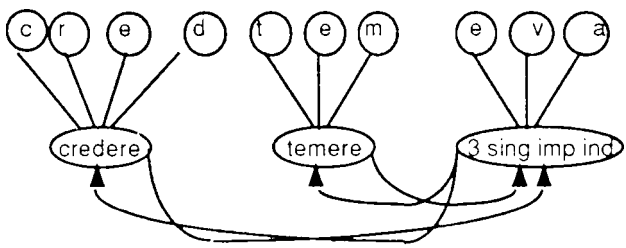


Fig. 3

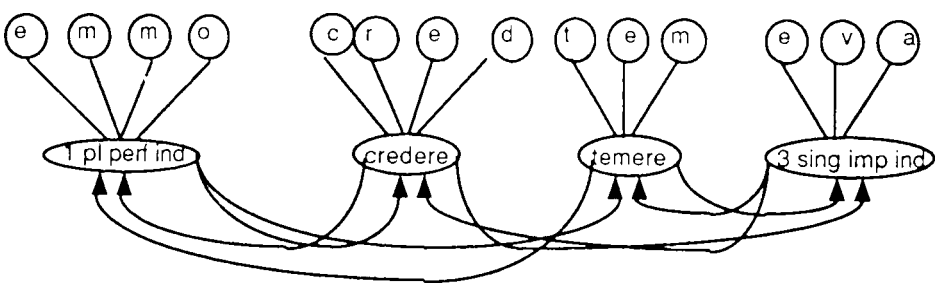


Table 2

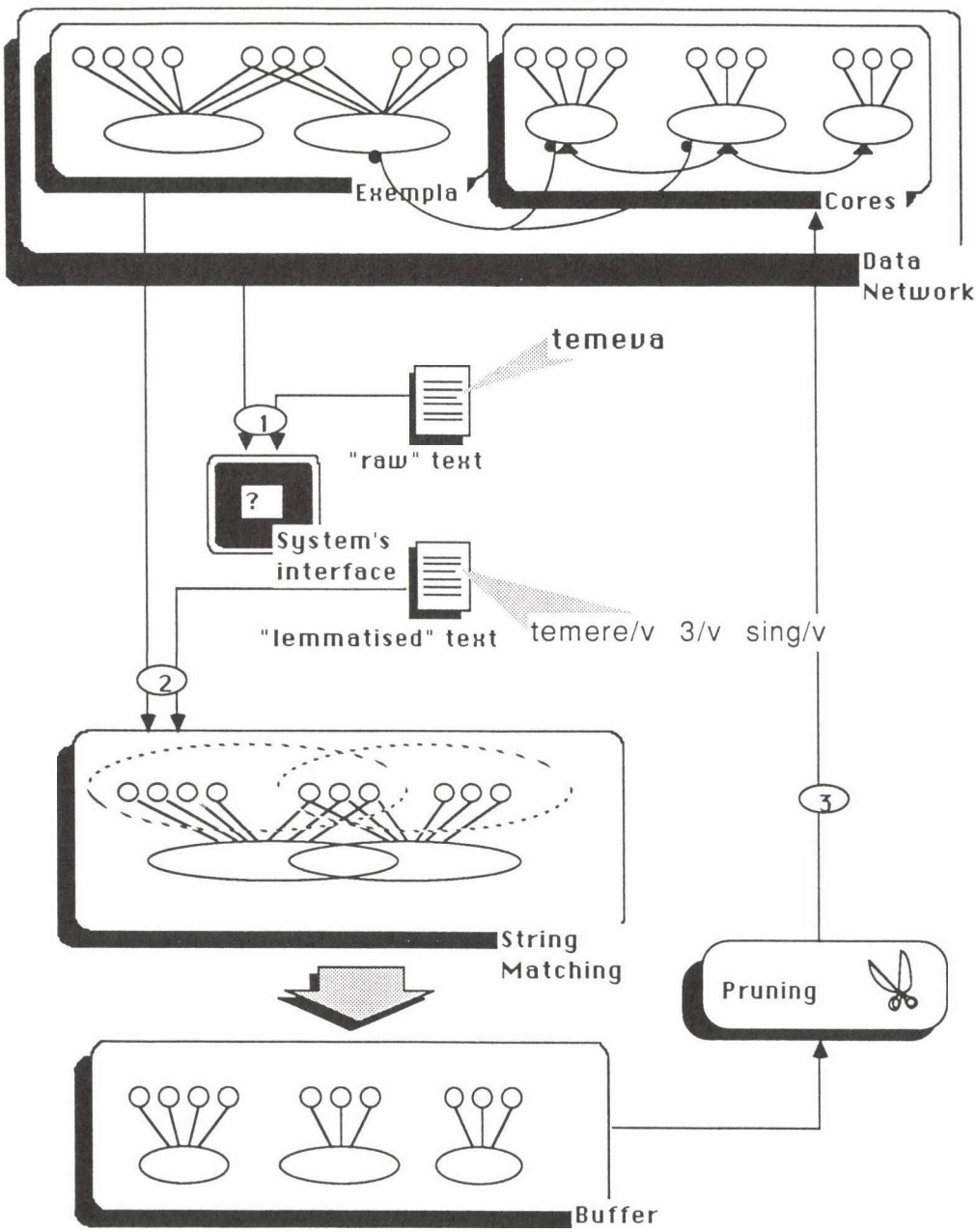
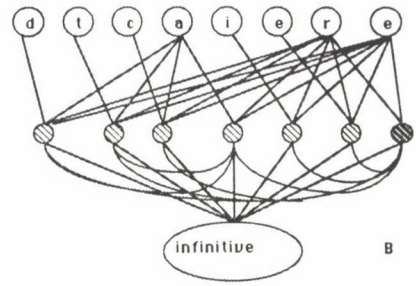

Exempla
Cores
Data
Network
temeva
1
"raw" text
?
System's
interface
"lemmatised" text
temere/v 3/v sing/v
2
3
String
Matching
Pruning
Buffer

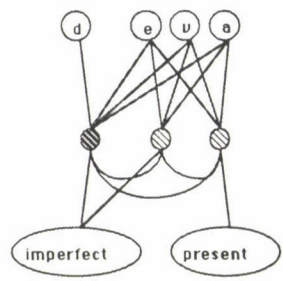
Table 3

A

rank	c.u.	rank	c.u.
18 -are	0%	25 -are	0%
46 -re	0%	71 -re	0%
47 -dare	17%	417 -dare	1%
48 -tare	23%	159 -tare	2%
104-iare	20%	454-iare	8%
182-lzare	3%	973-nare	10 %
190-ere	0%	135-ere	0%
239-care	0%	1367-vare	4%
259-emere	3%	1160-ttere	5%
270-icare	3%	657-icare	3%
284-rare	6%	222-rare	8%
338-tere	3%	687-tere	1%
343-giare	3%	1256-giare	1%
396-gere	17%	440-gere	5%
435-ndere	3%	600-ndere	1%
		251-ire	11%
COMPLEX		NEWS	



B



C

Table 4

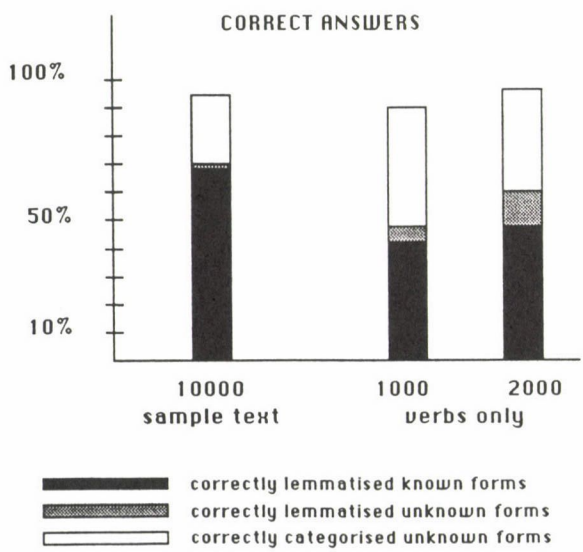


Table 5

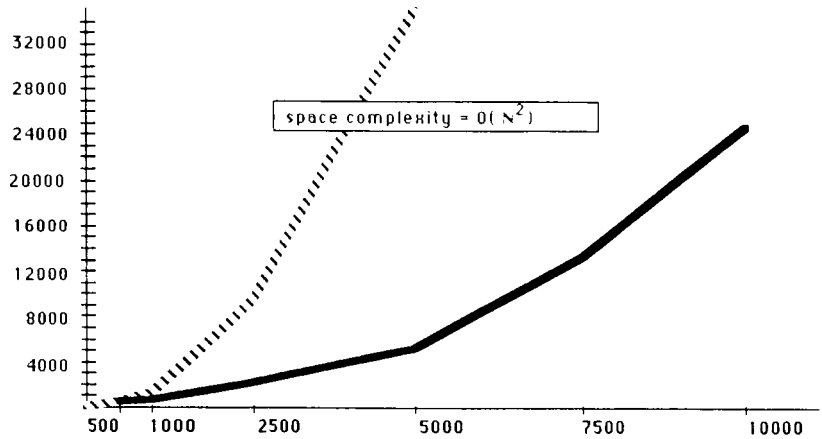


Table 6

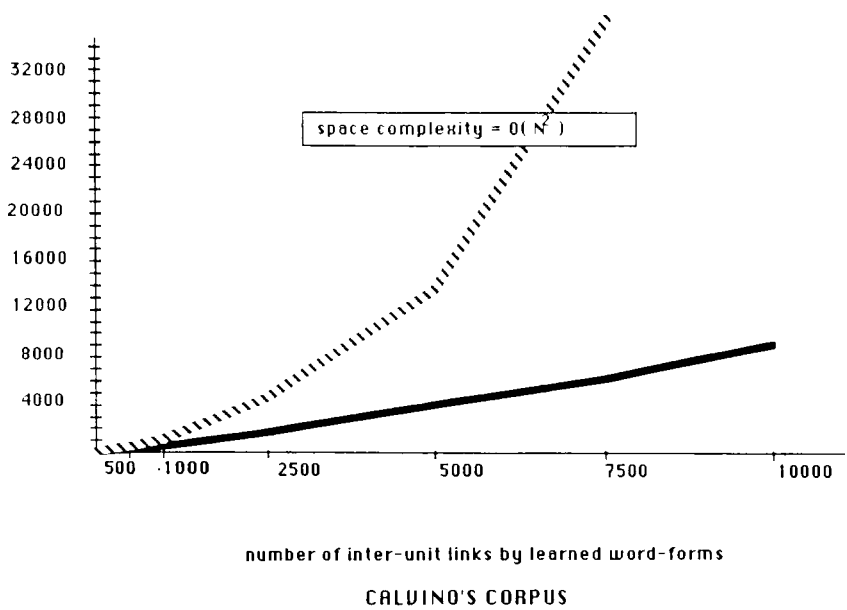
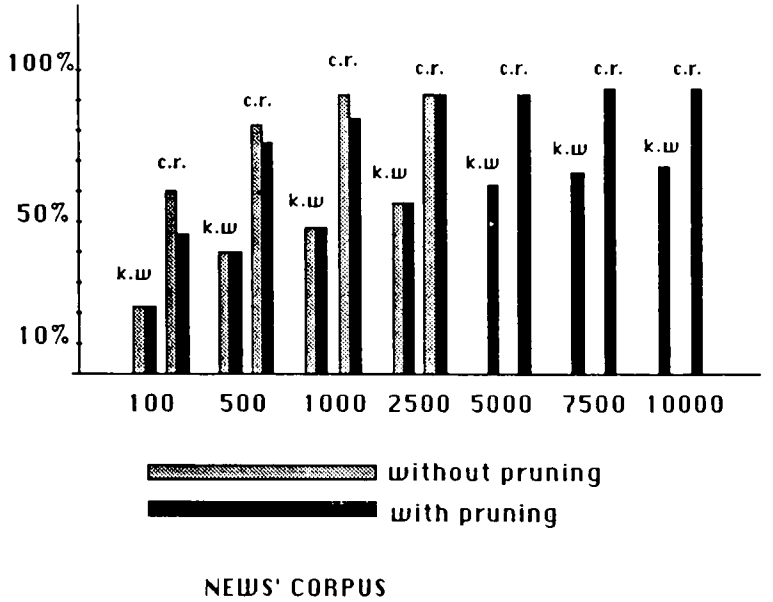
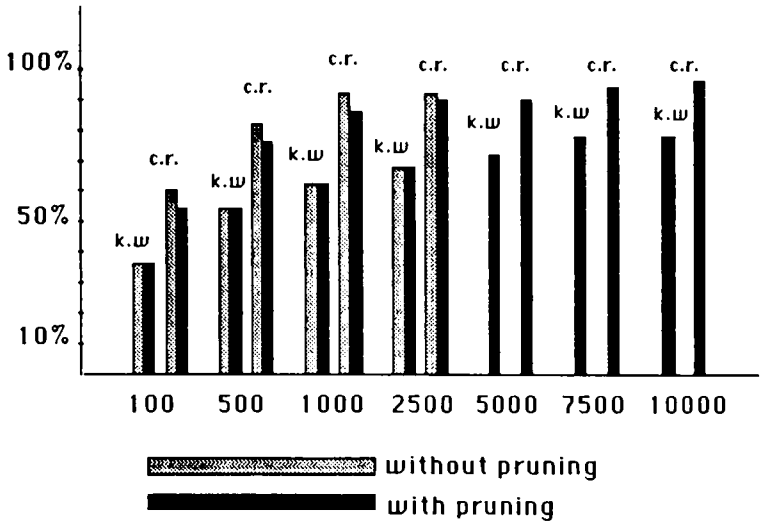


Table 7



NEWS' CORPUS

Table 8



CALVINO'S CORPUS

Table 9

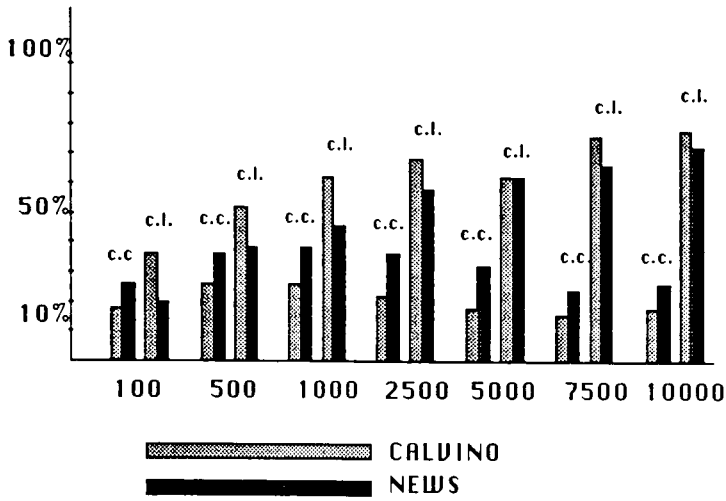
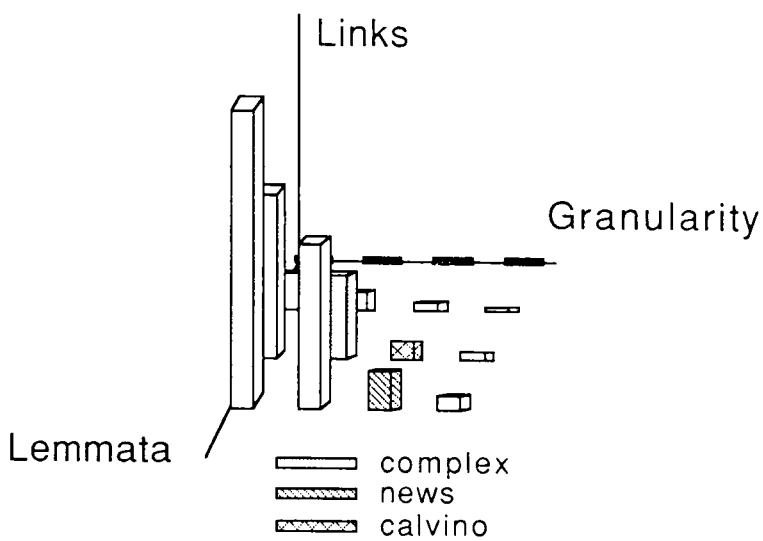


Table 10



Addresses of the authors: Vito Pirrelli
ILC-CNR
56100 Pisa
Italy

Stefano Federici
Par.O.LA. S.n.c.
56100 Pisa
Italy

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 265-275 (1992-93)

ACQUIRING AND REPRESENTING SEMANTIC INFORMATION FROM PLACE TAXONOMIES

ADRIANA ROVENTINI

1. Introduction

Place is a very general concept, and, in any dictionary, various meanings will be associated with it and many words belonging to different lexical fields will be put in relation with one or more of them. In the dictionaries we are considering: the Italian Machine Dictionary and the Garzanti Italian Dictionary (henceforth DMI and GRZ), the first sense of *luogo*, which is the best Italian equivalent for the English place, is:

1) *parte di spazio idealmente o materialmente delimitata*
   part of space ideally or materially delimitated

from which it is easy to deduce that, because any experience or object we can think or speak about exists in some ideal or materially delimited space, the concept of place will spread out over the dictionary in many different directions and will belong, as an attribute, to many different taxonomies. The other main word senses given in our dictionaries are the following:

2)	*parte della superficie terrestre*	part of the earth's surface
3)	*edificio o parte di esso*	building or part of it
4)	*parte di un oggetto, punto*	part of an object, point
5)	*passo di uno scritto*	written passage
6)	*momento opportuno*	right moment

To which the DMI adds:

7)	*luogo geometrico*	locus
8)	*condizione sociale*	status

Akadémiai Kiadó, Budapest

This set of meanings evidences the two essential dichotomies regarding the concept of place: the first opposing ideal or abstract places to material or concrete ones, the second opposing place as the typical object of the physical and geographical investigation and place as the theater of human activity as well as the repository of all human artifacts. In order to limit this investigation within such a broad and complex lexical domain, in this paper we will consider a set of places in which we find an intersection between the lexical types of both place and food, while the whole set of word senses defined as place is outlined only in its more general features. This selection is due to the fact that food is the lexical subset experimented for representation in the LKB within the ACQUILEX project.

In the dictionaries we analyzed *luogo* as genus term, selects 668 word senses from the DMI and 278 word senses from the GRZ (where the difference in number can be attributed to the difference in size) and, in order to retrieve the most general associated features, this first level of hyponyms was considered sufficient. Using the set of definitions as a corpus, semantic information, usually placed in the differentia part or differentia specifica, was extracted from them. This kind of analysis was essential in order to capture the different semantic features concerning the principal meaning types of place which could be represented in a more explicit and structured way either into templates following the example given by Calzolari (1991) for the concept of substance, or, when possible, by means of lexical rules as proposed by Briscoe–Copestake (1991a, b). They present a definition of lexical rule formalized within the framework of unification based approaches to the lexicon which employ typed features structures and default inheritance. They also argue that, in the lexicon, sense extension and morphological processes of derivation and conversion are similar enough to be treated as lexical rules and that the distinction between these phenomena is not as sharp as usually meant. On this purpose, the following phenomena were considered, when analyzing the metalanguage of the definitions:

a) the nouns connected with the genus;
b) the types of functions which link the genus to the differentia;
c) the properties that, in the differentia, are associated with it;
d) the connections between morphological and semantic features.

Within the corpus of the definitions, almost any of these phenomena are related to typical formulae or syntactic structures.

2. Types of place and sense extension

A typical procedure adopted by lexicographic metalanguage is to join the genus with another noun which is either an instantiation or an extension of it. This phenomenon is due to the necessity of compressing the information in printed dictionaries, from which it descends that, within a same definition, the instantiation of a meaning type is usually evidenced by a disjunction or by simple juxtaposition, while the sense extension is rather revealed by conjunction, as is exemplified below:

(a) *bassofondo: luogo o quartiere della malavita*
slum: place or neighbourhood of criminal underworld

(b) *galera: luogo e situazione in cui si soffre*
gaol: place and situation in which one suffers

In example (a) is given a first and more general indication, *luogo*, followed by an instantiation or specification of it *quartiere* (neighbourhood). In the second example (b), what is given is not a type of place but a metonymic extension of it: the condition of being in a certain place, in this case the condition of being in prison. Unfortunately, owing to the lack of consistency of the metalanguage, this use of conjunction or disjunction is unreliable; it was anyhow useful as cue of a typical phenomenon very frequent in this subset and worthy of a punctual analysis which pointed out the most common meaning types as well as the cases of sense extension by metonymy. The following are the cases of this kind of sense extension found in the whole subset of definitions:

luogo e azione (place and act of) as in: *attracco*, *imbarco* (docking, embarcation)
luogo e modo (place and way) as in: *collocazione* (collocation)
luogo e collezione (place and set of things) as in: *utensileria* (tool room)
luogo e persone (place and set of people) as in: *curia*, *cantoria* (curia, choir)
luogo e arte (place and art of) as in: *carpenteria*, *pasticceria* (carpentry, confectionery)
luogo, carica e durata (place, office and tenure) as in: *assessorato* (councillorship)
luogo coltivato e le piante (place and set of trees) as in: *oliveto* (olive grove)
luogo e quantita' (place and quantity) as in: *fungaia* (mushroom bed)
luogo, mobile e insieme (place, furniture and set of) as in: *guardaroba* (wardrobe)

luogo e tempo (place and time) as in: *esilio* (exile)
luogo e situazione (place and situation) as in: *galera* (gaol)
luogo e animali (place and set of animals) as in: *stalla*, *porcile* (cowshed, pigsty)

It is possible to add some more combinations but, as far as semantic information is concerned, these are sufficient to evidence a lexical phenomenon which is relevant from the point of view of its encoding within a lexical component for Natural Language Processing systems because of the change of the subcategorization features involved when shifting from concrete to abstract (e.g. place/art), from inanimate to animate (e.g. place/plants, place/animals), or from inanimate to human (e.g. place/people).

With regard to sense extension, it is interesting to note that certain types of places, for example those in which animals live or are kept and raised, in addition to the metonymic, show the metaphoric sense extension. In fact the metaphoric extension animal/human, evidenced by Briscoe–Copestake (1991b) is also productive within this lexical type. Words like *pollaio* (poultry pen), *stalla* (cowshed), *porcile* (pigsty), *nido* (nest), *formicaio* (anthill), *alveare* (beehive), *tana* (den) and so on, can be used to denote places inhabited or crowded by human beings and, with the same type of association, we can say 'This room is a pigsty' just as we say 'Luigi is a pig'. In the subset considered this phenomenon is regular enough to be dealt with by a lexical rule of the kind animal metaphor described in the above cited paper.

3. Functions, properties and constituents

When we went to analyze the differentia in our set of definitions these types of semantic information were considered relevant:

a) the aim or goal to which a place is devoted;
b) the properties which outline it in its appearance or formal side;
c) the constituents: items, structures, plants, animals which characterize it.

All this semantic information, in the general template for place (Fig. 1), is gathered under three main attribute tags: function, property, constituency, which, in a more general and comprehensive way, respectively correspond to the telic, formal and constitutive roles described by Pustejovsky (1991).

Function	
Addetto a	Intended for
Adibito a	Used as
Assegnato a	Assigned to
Attrezzato a	Equipped with
Destinato a	Intended for
Designato per	Appointed for
Coltivato a	Cultivated with
Piantato a	Planted with
Che serve a	Necessary for
Che + VP	Which + VP
Di + NP	Of + NP
Dove si + VP	Where + VP
In cui si + VP	In which + VP
Ove si + VP	Where + VP
Per + NP	For + NP
Da cui + VP	From which + VP
Donde si + VP	From Where + VP
Attraverso il quale si + VP	Through which + VP
Per cui si + VP	" "
Per dove si + VP	" "
Dove si + VP	Where + VP

Property	
Larghezza	Width
Altezza	Height
Profondita'	Depth
Forma	Shape
Struttura	Structure
Popolamento	Population
Posizione	Position
Esposizione	Exposure
Clima	Climate
Temperatura	Temperature
Umidita'	Humidity

Fig. 1
TEMPLATE for PLACE (and related defining patterns)

Coltivazione	Cultivation
Costruzione	Building
Apparenza	Appearance
	Constituency
Ricco di	Rich in
Pieno di	Full of
Folto di	Thick full of
Coperto di	Covered with

Fig. 1
(cont.)

The pure locative function is obviously inherited from the genus and is a basic value within the entire set of definitions. This is joined, in most of them, with telic functions evidencing the particular aim or goal to which a place is devoted. The metalanguage expresses the telos in different ways ranging from a simple adverb or preposition to more complex prepositional phrases, as appears in the examples below:

1a)	*luogo dove si allevano*	place where . . . are raised
1b)	*luogo in cui si allevano*	place in which . . . are raised
1c)	*luogo ove si allevano*	place where . . . are raised
1d)	*luogo di allevamento*	place of rearing
1e)	*luogo per l'allevamento*	place for rearing
2a)	*luogo adibito alla raccolta*	place assigned for gathering
2b)	*luogo di raccolta*	place of gathering
2c)	*luogo ove si raccolgono*	place where . . . are gathered

In these cases one telic tag will be able to substitute different but equivalent formulae. The same is worth for the purposes which have a same semantic value (the rearing or the gathering) but are defined using either a verb phrase as in 1a, 1b, 1c, 2c, or a noun phrase formed by the corresponding deverbal noun as in 1d, 1e, 2a and 2b.

A second type of function frequently combined with place is that of motion. Several places have the character of being reference marks with this regard. There are places from which, or towards which, or across which the motion is directed, and all these are characterized by the joint locative-motion

function. Most of these typical linking formulae are listed in the Template for Place (Fig. 1) under function attribute.

As regards the properties associated with place most of them come from definitions which are very simple in their structure and usually consist of the genus plus one or more modifiers: adjectives or past participles with an adjectival value. The following are a few examples of this kind of definitions where the genus is described in its purely formal appearance:

luogo caldo	warm place
luogo squallido e sporco	dreary and dirty place
luogo abitato	inhabited place
luogo montuoso	mountainous place

By means of a careful analysis of the whole set it was possible to acquire a number of this kind of semantic features and to gather them using a few tags under the generic attribute property. Obviously not all the modifiers fall into one of these features, but certainly most of them do.

An example of a typical formula revealing the constitutive argument of a noun, constituency in the Template, is shown in the definition below:

sterpaia (scrub):	*luogo pieno disterpi* (place full of thorns)
sassaia (stony place):	*luogo pieno di sassi* (place full of stones).

4. Lexical regularities and morphological features

The hyponyms of place show some interesting regularities either in their word formation or in their lexical features. The suffixes which form homogeneous sets of words denoting places are the following:

-eto	*-eta*	
-aio	*-aia*	
-oio	*-ile*	
-eria	*-oria*	
-tura	*-ario*	and also
-ficio	*-coltura*	

as second element of many compounds.

These sets mostly refer to particular types of places which exhibit these defining patterns:

il luogo coltivato e le piante	the cultivated place and the plants
luogo coltivato a/ad	cultivated place for
luogo dove si allevano	place where . . . are/is raised
luogo dove si conserva/no	place where . . . are/is kept
luogo dove si fa o si vende	place where . . . are/is made or sold
luogo folto di	place thick with . . .
luogo in cui crescono	place where . . . grow
luogo ricco di	place rich in
luogo piantato a	place with a plantation of . . .
luogo pieno di	place full of

Because these formulae define words denoting places where:

a) edible plants are cultivated or spontaneously grow such as land, field, wood, plantation;

b) animals are grown such as rearing, breeding, farm;

c) animals or vegetables are transformed into food, such as factory, laboratory;

d) food is eaten and/or sold;

the search was deepen in this direction in order to select homogeneous subsets of words which could be put in relation with lexical types of food in the LKB. Among these hyponyms we choose to outline the template of *terreno* (land) in order to show a set of features representing a subtaxonomy of place (Fig. 2).

	Function
Adibito a	Used as
Destinato a	Intended for
Pronto per	Ready for
Adatto a	Suitable for
Coltivato a	Cultivated with
Lavorato a	Cultivated with
Piantato a	Planted with
Tenuto a	" "

Fig. 2
TEMPLATE for TERRENO (and related defining patterns)

	Property
Esposizione	Exposure
Struttura	Structure
Altezza	Height
Forma	Shape
Ampiezza	Width
Posizione	Position
Coltivazione	Cultivation
Umidita'	Humidity

	Constituency
Coperto di	Covered with
Folto di	Dense with
Pieno di	Full of
Ricco di	Rich in
Ricoperto di	Covered with

Fig. 2
(cont.)

4.1. Lands, fields, woods, farms and cultivated areas

These types of places are characterized by the following suffixes: *-eto/a*, *-aio/a*. A few examples are: *castagneto*, *acereta*, *oliveto*, *asparagiaia*, *carciofaia*, *zuccaia* (chestnut wood, maple wood, olive grove, asparagus field, artichoke field, pumpkin field). The suffix *-eto/a* is the most interesting because it carries an univocal meaning and could be dealt with by a lexical rule which specifies that every noun of tree, plus the suffix *-eto* (in a few cases *-eta*), forms another noun denoting either the place or, by extension, the set of plants growing there.

The other suffix works in the same way in this subset, but also forms words denoting place + animals such as: *colombaia*, *pollaio* (dovecot, poultry pen), and place + quantity or set of materials such as *legnaia*, *ghiacciaia*, *granaio* (wood store, icebox, granary). Furthermore it enters into the formation of nomina agentis such as *macellaio*, *fornaio*, *lattaio* (butcher, baker, milkman).

Two other sets of words homogeneous in meaning and word formation are constituted by hyponyms of *piantagione e allevamento* (plantation and breeding). Both use the noun *coltura* as the second element of compounds denoting cultivations or rearings from a technical point of view and all these hyponyms are regularly defined as follows:

a) *apicoltura: allevamento delle api* (the keeping of bees)
b) *viticoltura: coltivazione della vite* (the growing of grapes)

The first word sense of both *allevamento* and *coltivazione* in Italian is 'art of . . . ' but they also denote the place in which the cultivation or the rearing are performed and the set of plants or animals grown. It derives that, in some contexts, the meaning of words like *apicoltura* can shifts to denote the place where the 'art' is carried on.

4.2 Factories and shops

The places in which various artifacts are made or sold are derived from nouns by means of the suffix *-eria* or, especially in the case of factories, we can find compounds in which the second element *-ficio* is the carrier of the univocal meaning: 'factory of . . . '. This second type of word formation is interesting because it is very productive and, as *-eto*, could be manageable by means of rules. On the other hand *-eria*, even if productive in this lexical field, as it produces many other different formations with regard to meaning, would be difficult to manage by lexical rules. In this subset it forms nouns denoting shops, and in some cases, by extension, the set of goods which there are sold and, in some cases, also the art of making such artifacts. An example of that is the noun *pasticceria* (confectionery) which shows all these types of meaning: 'place', 'art of' and 'set of'.

5. Final remarks

It is interesting to highlight that most of the suffixes which form nouns denoting places (and extensions of them) do not give rise to changes in grammatical categories. They derive denominal nouns in which only the meaning is involved in the change. For this reason, they could be seen as a kind of marked sense extension and, in some cases, be handled by lexical rules (see the cases of *-eto* and *-ficio*); nevertheless, given that most of them do not carry univocal meanings but, on the contrary, derive various types of word senses and are productive in many different lexical domains, in our opinion it is not convenient to manage them by lexical rules. By means of lexical rules, instead, should be treated a phenomenon as sense extension which, as shown in this paper, is very frequent throughout the various levels of the place taxonomy. Frequent, in the subsets analyzed, is the case in which an inanimate noun denoting a place, is used in an animate, collective sense denoting a group found within it. In these cases it is worth finding a lexical rule for the general sense extension

inanimate noun—animate collective noun and domain-specific sub-rules could then be set up to capture the various types within this category.

References

Boguraev, B.K.–Briscoe, E.–Calzolari, N.–Cater, A.–Meijs, W.–Zampolli, A. 1988. Acquisition of Lexical Knowledge for Natural Language Processing System, proposal for ESPRIT BRA, Cambridge.

Briscoe, E.–Copestake, A. 1991. Sense extension as lexical rule. In: Proceedings of the IJCAI Workshop on Computational Approaches to Non-Literal Language. Sidney, Australia. ESPRIT BRA-3030 ACQUILEX Working Paper No. 022.

Calzolari, N. 1991. Representation of semantic information in a Lexical Knowledge Base. In: Proceedings of the ACL SIGLEX Workshop on Lexical Semantics and Knowledge Representation, 188–97. Berkeley, CA.

Copestake, A.–Briscoe, E. 1991. Lexical operations in a unification-based framework proceedings of the ACL SIGLEX Workshop on Lexical Semantics and Knowledge Representation, 88–101. Berkeley, CA. ESPRIT BRA-3030 ACQUILEX Working Paper No. 021.

Ostling, A. 1991. Sense extensions in the Italian food subset. ACQUILEX, ESPRIT BRA-3030, Working Paper No. 024, ILC-December–1991, 88–101. Pisa. ESPRIT BRA-3030 ACQUILEX Working Paper No. 021.

Pustejovsky, J. 1991. The generative lexicon. In: Computational Linguistics 17(4).

Address of the author: Adriana Roventini
Istituto di Linguistica Computazionale, C.N.R.
Via della Faggiola 32
56100 Pisa
Italy

Acta Linguistica Hungarica, Vol. 41 (1-4), pp. 277-306 (1992-93)

PAT EXPRESSIONS: AN ALGEBRA FOR TEXT SEARCH

AIRI SALMINEN–FRANK W. TOMPA

1. Introduction

Text search operations are used to locate and retrieve needed information from some text collection. In traditional information retrieval, text search is a means for identifying relevant documents (Salton–McGill 1983; Lee 1985). By specifying selection criteria for the text of a document, the reader can choose a subset of a given set of documents. If the text collection is defined not as a set of documents, but more generally as a structure containing some parts, then text search involves the specification of those parts of interest to the reader. The structure of the documents may be determined by the search system, by the author, by the text installer, or by the reader.

In the PAT™ system (Gonnet 1987; OTC 1993)[1] text search operations are expressions that efficiently combine traditional search capabilities with some new, powerful facilities. PAT contains means for lexical search, proximity search, contextual search and Boolean search (Hollaar 1979; Larson 1984; Lee 1985; Burkowski 1991). It also contains more rare operation types, including position and frequency search. Furthermore, a novel feature in PAT is the capability by which a reader can define structures for a text and use these structures in subsequent operations. One of the goals of this paper is to introduce the powerful search capabilities of PAT expressions.

Text search is usually considered so simple that only a rough description of the operations is given. For example, when word search is discussed, we are seldom told what is meant by a "word". The reader has to find out through experimentation how many words are contained in the strings "Jean-Marie" and "O'Hara". However, a careless description of search operations may lead to search errors or unnecessarily long retrieval sessions. A second goal of the paper, therefore, is to introduce a mechanism for precise specification of text search semantics.

[1] PAT is a registered trademark of Open Text Corporation.

Akadémiai Kiadó, Budapest

Text search using PAT is typically simple and straightforward (Raymond–Fawcett 1990). However, because of the powerful definition capabilities included in PAT, explaining and understanding the semantics of some operations may be difficult. As a side-effect of our systematic specification of PAT, we have identified some features of PAT expressions that cause problems and thus would benefit from further development. From this we see that precise specification also serves as a means for evaluation and offers a means for comparing text search systems.

As is common in information retrieval systems, a PAT search is applied to indexed text (see, for example, Gonnet 1983; Croft 1984; Larson 1984; Faloutsos 1985; Salton 1989; Burkowski 1991). Indexing is usually described from the point of view of implementation, for example, by giving an algorithm for the indexing (Salton 1981; Salton 1989; Gonnet 1991). However, since the way text is indexed affects search behaviour, our systematic approach to precise description must include mechanisms that accommodate indexing definition capabilities.

2. The PAT system

PAT is a text searching system developed at the University of Waterloo's Centre for the New Oxford English Dictionary and Text Research (Berg *et al.* 1991; Tompa 1992), and commercially available from Open Text Corporation in Waterloo (OTC 1989–93). PAT is used, for example, for searching the 570 megabyte *Oxford English Dictionary*, among other texts and text collections. As distributed, the PAT system has two different end-user interfaces: one based on a command language using PAT expressions (Fig. 1), the other based on direct manipulation interaction in which the system creates PAT expressions from the point-and-click actions of the end user (OTC 1993) (Fig. 2). PAT also provides an applications programmers' interface by means of which customized front ends can be written, communicating with the search engine via PAT expressions.

The text search is based on three innovative techniques: PAT indexing, region definitions, and PAT expressions. PAT accesses the original text with the aid of an associated index (Gonnet *et al.* 1991) that maps index terms to their occurrences in the text, which we shall call *indexed elements*. The choice of which text segments to index in PAT is not fixed by the system. Instead, PAT's indexing offers the text installer the capability to define the indexed elements of the text. The indexed elements may consist of, for example, all individual characters of the text, or multicharacter words separated by delimiters. Each

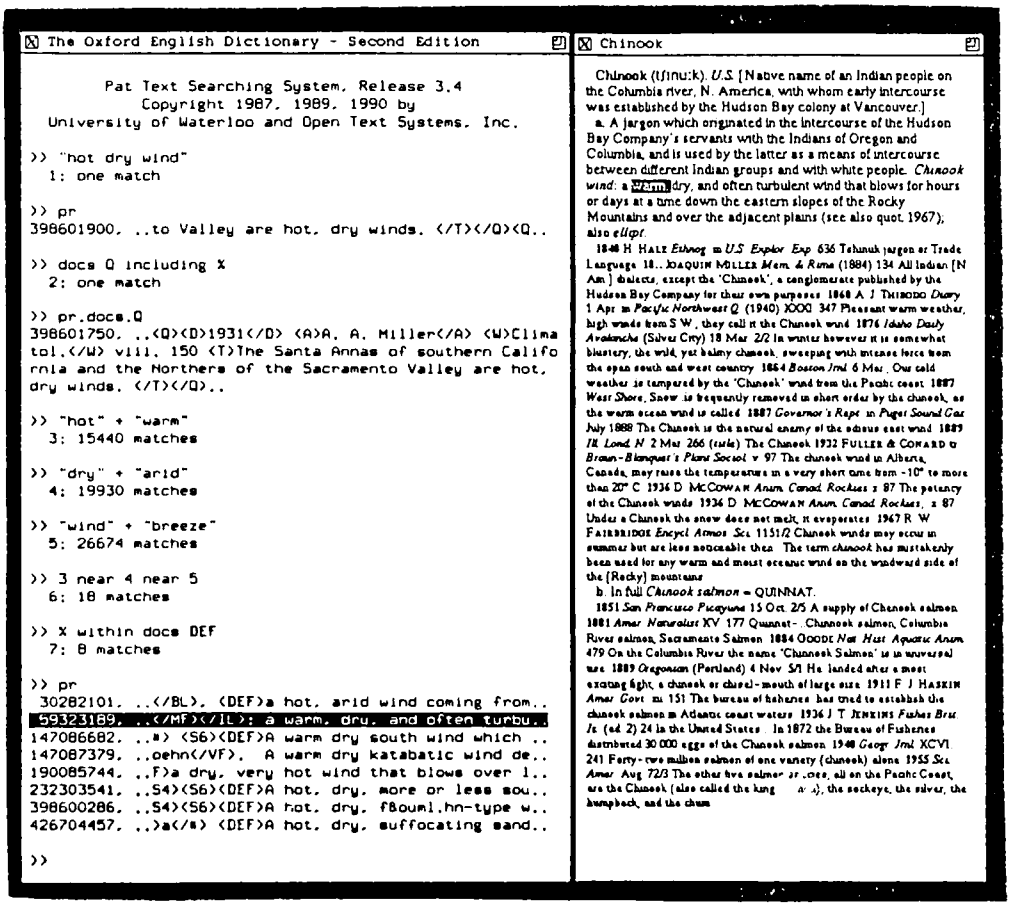


Fig. 1
Command-line interface with corresponding display window

indexed element is considered to be the start of a "semi-infinite string" that continues to the end of the text. In processing a query, the system finds those characters that begin semi-infinite strings matching the string given by the user. The text installer defines how the matching is determined. It is as simple to find single indexed elements or their prefixes as it is to find longer phrases, consisting of several indexed elements. If all characters are defined as indexed elements, the search in an indexed text corresponds to truly full-text search (where "the" matches the second character in "other").

In response to a query, PAT returns a result set, which is either the set of *match points* or the set of *regions* satisfying a given criterion, expressed by a

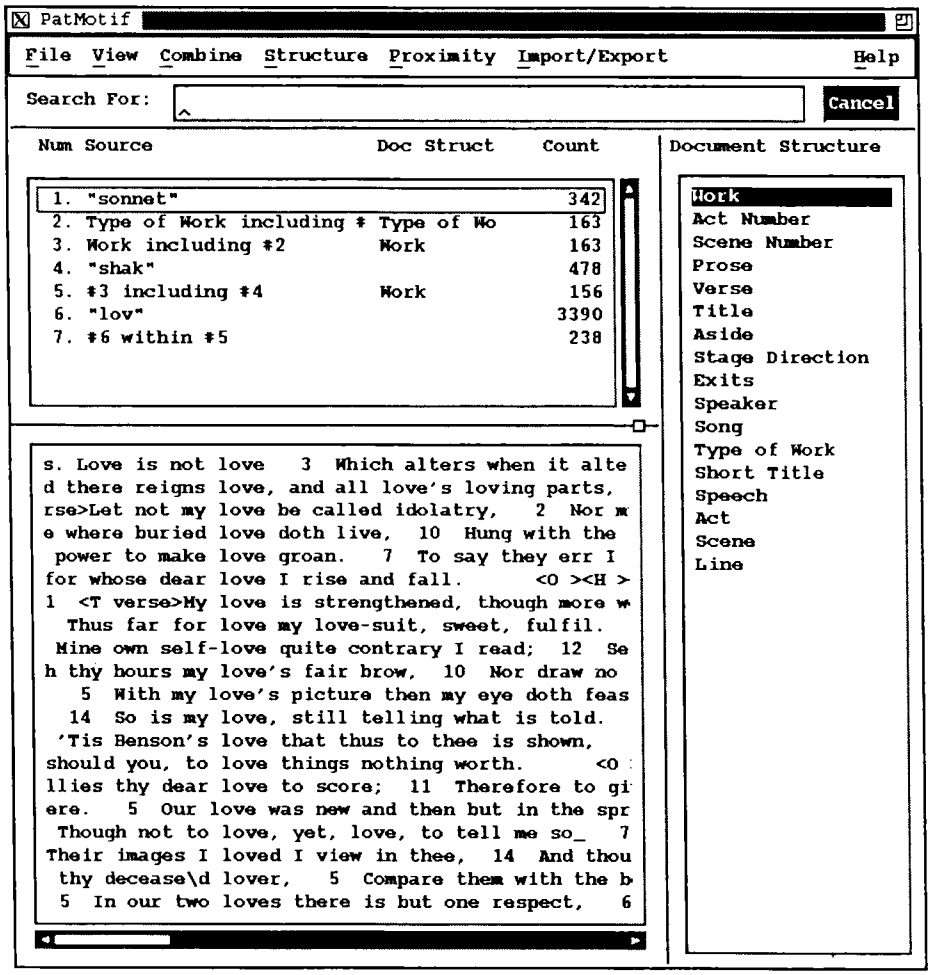


Fig. 2
Point-and-click interface to PAT

PAT expression. A match point is a character (starting a semi-infinite string), whereas a region is a substring of the text beginning and ending with specified characters. Regions are defined either by the text installer or by the reader. If the text is tagged (e.g., using SGML's reference syntax cf. Goldfarb 1990), the regions with a given name can be defined to begin and end with given tags. Alternatively, any characters found by text search can be used to denote the starts and ends of regions.

Result sets from one PAT expression can be used as operands for subsequent expressions. By default, a search command shows the count of match points or regions in the result set. If the result set is a match point set, characters from the left and right context of the match points in the set can be printed by a printing command. If the result set is a region set, the reader can print either regions from the set or characters from the left and right context of the beginning characters of the regions. In a workstation environment the text of a region may be displayed in a formatted form in a LECTOR™ window as in Fig. 1 (Raymond 1992). [2]

3. Text in PAT

In earlier work, grammars were used to describe indexing as a view of the original text, and this view was then further used to specify text retrieval operations (Tague *et al.* 1991). In this paper we will use the same approach, based on the text model introduced in (Salminen–Tompa 1992), which extends the earlier grammar-based model of (Gonnet–Tompa 1987).

3.1. Text as a character string

Any PAT text can be viewed as a sequence of characters. The following grammar, consisting of two productions, can be used as the schema for any such text:

(S) **string ::= character+.**
character ::= 'a' | 'b' |... .

The grammar defines two text *types*: **string** and **character**. The first production indicates that a string consists of one or more characters. (The symbol + in the production denotes iteration one or more times.) Thus, a *value* of type **string** is any nonempty sequence of values of type **character**. The second production enumerates all available values of type **character**. (The symbol | separates alternatives.)

Consider the following sample text:

```
<h>Consumer spending in U.S. up 1.5 per cent in June</h>
```

When we use this text as a text context for the schema (S), it is viewed as having been parsed by the grammar, and we can refer to *parts* of the text. Each

[2] LECTOR is a registered trademark of Open Text Corporation.

part is a text entity corresponding to one of the defined types. The sample text contains one part of type **string**, i.e. the whole text, and 56 parts of type **character**. The value of the **string** part is the complete sample text above. The value of the first **character** part is "<", the value of the second **character** part is "h", the value of the 49th **character** part is "J", and so on. PAT capitalizes on the fact that each **character** part is identified with a unique position in the whole text.

3.2. Text as an indexed string

Alternatively, PAT text can be viewed as a string consisting of indexed elements and delimiters separating indexed elements. Each indexed element begins a new phrase, continuing to the right from the indexed element until the end of the text. Therefore, if the string contains n indexed elements, it also contains n such phrases. For the above sample text, if (for simplicity) each contiguous sequence of non-blank characters is an indexed element and blanks are delimiters, there are ten phrases corresponding to the ten indexed elements:

```
<h>Consumer spending in U.S. up 1.5 per cent in June</h>
spending in U.S. up 1.5 per cent in June</h>
in U.S. up 1.5 per cent in June</h>
U.S. up 1.5 per cent in June</h>
up 1.5 per cent in June</h>
1.5 per cent in June</h>
per cent in June</h>
cent in June</h>
in June</h>
June</h>
```

We can define this formally using the following rules:

(a) **string ::= [delimiter] phrase.**
(b) **phrase ::= indexed_element [delimiter] [phrase].**

The first production indicates that a string may be a phrase or it consists of a delimiter followed by a phrase. (The square brackets denote optionality.) The second production shows that a phrase always begins with an indexed element, which may be followed by a delimiter and then by another phrase.

Thus, two indexed elements may be separated by a delimiter, or one indexed element may follow another directly.

Indexed elements and delimiters are described specifically for each text file by the text installer. For most text retrieval applications, text installers choose indexed elements to correspond as closely as possible to the users' perception of "words". Figure 3 shows a possible grammar describing the indexing of a text containing standard markup tags (e.g., this grammar describes the indexing used for the *Oxford English Dictionary* at the University of Waterloo and elsewhere).

```
(1)  string ::= [delimiter] phrase.
(2)  phrase ::= indexed_element [delimiter]
                    [phrase {where indexed_element
                                 {is preemptive_element or preceded by delimiter} }].
(3)  indexed_element ::= preemptive_element | limited_element.

(4)  preemptive_element ::= stand_alone_char | signal_char element_char*.
(5)  limited_element ::= element_char+.
(6)  delimiter ::= delim_char+.

(7)  stand_alone_char ::= hyphen.
(8)  signal_char ::= less | ampersand.
(9)  element_char ::= a |... | z | A |... | Z | d1 |... | d9 | hash | slash.
(10) delim_char ::= blank | period | comma | greater | colon | apostrophe |....

(11) hyphen ::= '-'.
     less ::= '<'.          ampersand ::= '&'.
     a ::= 'a'.        ...  z ::= 'z'.
     A ::= 'A'.        ...  Z ::= 'Z'.
     d0 ::= '0'.       ...  d9 ::= '9'.
     hash ::= '#'.          slash ::= '/'.
     blank ::= ' '.         period ::= '.'.        comma ::= ','.
     greater ::= '>' .      colon ::= ':'.         apostrophe ::= "'.....
```

Fig. 3
A grammar describing an indexed text

The grammar defines one way to divide a string into indexed elements and delimiters. So that an indexed element can be easily recognized syntactically, it must either begin with a specific character (**preemptive_element**) or it must be preceded by a delimiter. This constraint is specified in production (2) by replacing the use of the simple non-terminal in (b) above by the "property" **phrase {where indexed_element {is preemptive_element or preceded by delimiter} }** (see Salminen (1992) for a detailed explanation of properties). Preemptive elements are either single characters (for this specific grammar, hyphens constitute the only such elements), or they start with a **signal_char**

(for this grammar, either a less-than-sign or an ampersand) and then continue with zero or more element characters (here any combination of letters, digits, hash-sign, or slash). Delimiters contain characters distinct from any of these characters. Productions (7)–(10) contain the text-specific assignment of characters to the four classes that define which substrings of the text are indexed elements. They use the nonterminals defined in productions starting from (11) so that character transformations can be separately specified for evaluating whether a given phrase matches a query (see section **4.1**). The effect of this particular choice of indexing is that most punctuation characters are treated as blanks (periods, commas, etc. are not part of any indexed elements).

Consider again the sample text above. As PAT text this is a sequence of characters, containing parts of types **string** and **character**. However, using Fig. 3 as an indexing description for the text means that the text is reparsed by the grammar. The text context then includes simultaneously parts having types from the schema (S) and parts having types from the indexing description. The text context contains 14 indexed elements which are shown below (the extent of each marked by |-----).

```
<h>Consumer spending in U.S. up 1.5 per cent in June</h>
|- |------- |------- |- | |  |- | | |-- |--- |- |---|--
```

Notice that delimiters are not contained in **indexed_element** parts. Furthermore, the two last indexed elements are not separated by a delimiter: the character '<' begins a new indexed element immediately.

The grammar described in Fig. 3 has been used most often for PAT applications. From the grammar it is clear that "Jean-Marie" contains three indexed elements and "O'Hara" contains two. An alternative index that makes *every* character an indexed element has been used to support detailed proofreading of the *Oxford English Dictionary* in preparation for the second edition. Between these two extremes exist many other possibilities: for example, specification and program text can be indexed on all upper case letters as well as conventional word starts so as to allow a software engineer to find all mention of functions and variables having to do with a "window" even if they are named "AdjustWindowImage" and so forth. Such indexing can be simply described through reassignment of upper case letters to **signal_char** instead of **element_char**.

This is not intended to describe the way PAT indexing is implemented (Gonnet *et al.* 1991). Instead, we describe the indexing from a reader's viewpoint, specifically, the effect of indexing on search operations. From Fig. 3, the

addressable units of a text are clearly identified. As opposed to the simplistic listing of phrases given above, it is precisely stated that the sample text with such an index has 14 retrievable phrases:

```
<h>Consumer spending in U.S. up 1.5 per cent in June</h>
Consumer spending in U.S. up 1.5 per cent in June</h>
spending in U.S. up 1.5 per cent in June</h>
in U.S. up 1.5 per cent in June</h>
U.S. up 1.5 per cent in June</h>
S. up 1.5 per cent in June</h>
up 1.5 per cent in June</h>
1.5 per cent in June</h>
5 per cent in June</h>
per cent in June</h>
cent in June</h>
in June</h>
June</h>
</h>
```

Because the indexing described in Fig. 3 includes period as a delimiter character, the strings "U.S." and "1.5" each contain two indexed elements. The current PAT indexing capability is very simple, and we cannot define context-dependent differences in the use of a character. Thus even though the strings "U.S." and "1.5" might seem more appropriately considered as one indexed element each, we cannot define them so without causing end-of-sentence periods to also be included in indexed elements.

Slashes may also cause some problems with this indexing. Because the indexing description is made for text with standard markup begin and end tags (for example, "<h>" and "</h>"), it is defined such that each begin and end tag contains one indexed element: for example, "<h" and "</h". (The character '>' ending a tag is defined to be a delimiter.) Slash must be defined as an indexed element character if the identifier within the end tag (e.g., the "h" within "</h>") is not to be made a separate indexed element. However, as a result the substring "USA/Canada" within a text context will also be treated as one indexed element. Subsequently, if the reader searches for occurrences of "Canada", the substring within "USA/Canada border" fails to match.

From these examples, we see that defining the indexing of text properly is somewhat problematic with PAT. By extending PAT's indexing definition capabilities, more flexibility for application-dependent indexing could be achieved.

However, whatever the indexing definition techniques, the richness of natural language and the variety of information needs from natural language text will always cause problems in defining indexing satisfactorily.

3.3. Regions

A region is a substring of PAT text, beginning and ending at specified characters. Each region belongs to one or more *region sets*. A region cannot overlap other regions belonging to the same region set, but it can overlap regions in other sets arbitrarily. This concept is thus a unification and generalization of the concepts of "document" and "field" used in conventional information retrieval systems.

From within PAT, a region set can be created by a region definition, which gives the condition determining the first and last characters of each component region. Region sets can also be derived from prior region sets. The form of the region definitions is described in section 4.4.

A region definition corresponds to a new grammar by which the text context can be reparsed, and through which new parts can be identified in the text. For example, for the following text we could define a region set, named "year", to include the two regions as indicated:

```
Fascicles of the OED appeared between 1884 and 1924.
                                      ----     ----
```

The capability to handle regions is the feature that most distinguishes PAT from conventional document retrieval systems. Regions are defined either by the text installer (pre-defined regions) or by the reader (user-defined regions), the latter serving as temporary definitions of scope or as personal "views" of the text for limiting queries or responses. It is primarily through judicious definition of region sets that texts can be structured to meet the needs of diverse applications.

4. PAT operations

PAT is a set-at-a-time algebra for manipulating results of text queries. Each PAT expression is either a match point expression, specifying a set of characters in the text context, or a region expression, specifying a set of regions in the text context. The sets specified by match point expressions or region expressions are called match point sets and region sets, respectively; collectively they are called *result sets*.

Result sets produced by search commands are numbered sequentially, and they can be referenced in subsequent expressions by number. A user can optionally assign a name to a result set via a search command, in which case the result set can be subsequently referenced by name. If e is a PAT expression, then a command of the form

$$n = e$$

gives the name $*n$ to the result set specified by e. If e is a match point expression, then $*n$ refers to the corresponding match point set; if e is a region expression, $*n$ names the corresponding region set. Finally, the symbol % refers to the immediately preceding result set.

The syntax of PAT expressions is described in the Appendix. PAT operations can be classified by type:

	search class	*result set type*
(1)	lexical search	match points
(2)	position search	match points
(3)	frequency search	match points
(4)	region definition	regions
(5)	restriction	match points / regions
(6)	augmentation	match points / regions

As indicated in the table, the resulting expressions in a class can be match point expressions or region expressions only, or both.

Expressions in classes (1), (2), and (3) are always match point expressions, i.e. they are used to search for characters. In lexical search the user searches for phrases by giving one or two patterns. The result set consists of the first characters in the found phrases. Position search means searching for a character in a given position in the whole text, or searching for characters at fixed offsets to the left or right of the match points of a given match point set. Frequency search means identifying match points for frequently appearing substrings or long repetitions from the text. Expressions from classes (1), (2), and (3) are discussed further in sections **4.1**, **4.2** and **4.3**, respectively.

Expressions in class (4) are always region expressions: a region definition specifies a region set as a function of two match point sets. Region definitions are discussed further in section **4.4**.

Expressions in classes (5) and (6) are either match point expressions or region expressions. These produce new result sets as a function of existing sets and are discussed in more detail in sections **4.5** and **4.6**, respectively.

4.1. Lexical search

A PAT expression for lexical search is either a character string or of the form $s_1 .. s_2$ where s_1 and s_2 are character strings. Lexical search always yields a match point set.

By giving a string s the reader searches for all characters in the text that begin phrases matching s. The matching of a phrase with a string pattern is determined after normalizing both the phrase and the pattern. The normalization inherent in PAT maps delimiter characters to blanks. However, concurrently with the indexing, the text installer may describe additional normalization, through which characters can be deleted or replaced by alternative characters.

In grammar based text modelling, normalization can be described as a text translation, defined by a set of translation productions that redefine those text types of the indexing description whose parts are changed by normalization. The translation productions, together with the indexing description, define a translation of a character string to a normalized string.

Figure 3 showed one possible indexing for a text. For such a text, normalization could be defined by a table showing replacement productions as follows:

indexing production	*normalization replacement*
string ::= [delimiter] phrase.	**string ::= phrase.**
delimiter ::= delim_char+.	**delimiter ::= ' '.**
A ::= 'A'.	**A ::= 'a'.**
...	...
Z ::= 'Z'.	**Z ::= 'z'.**

The replacement for the **string**-production indicates that if the string begins with a delimiter, the delimiter is removed in normalization. The **delimiter**-production causes the replacement of all delimiters by blanks, and the rest of the productions define that each upper case letter is replaced by the corresponding lower case letter. The effect of these replacements is that pattern matching is case-insensitive, delimiters cannot be distinguished, and the presence or absence of a delimiter before the start of a match cannot be determined.

As described in section **3.2**, PAT allows a text installer to define the partitioning of characters into the classes **stand_alone_char**, **signal_char**, **element_char** and **delim_char** to customize indexing. The text installer customizes normalization by assigning replacement characters as well, under the constraint that replacements must be chosen from the same class as the character being replaced (e.g., lower case letters can be substituted for upper case letters only if both are in the same class, in this case **element_char**). There is also a limited facility for defining stopwords, that is, specific character strings that are to be treated as delimiters rather than as indexed elements. Extensions to the language are being developed to generalize these capabilities by allowing a text installer to define the indexing and normalization transductions using a powerful language for describing Mealy machines (Gonnet–Snider 1992). In the examples of the rest of the paper we assume that the indexing is defined by the grammar in Fig. 3 and the normalization by the translation productions given above.

A phrase in the text matches a given string if the normalized string is a prefix of the normalized phrase. Consider our earlier sample text:

```
<h>Consumer spending in U.S. up 1.5 per cent in June</h>
```

The string "in" matches two phrases:

```
in U.S. up 1.5 per cent in June</h>
in June</h>
```

The string "s" also matches two phrases:

```
spending in U.S. up 1.5 per cent in June</h>
S. up 1.5 per cent in June</h>
```

because through normalization the sample text is translated to the form

```
<h consumer spending in u s up 1 5 per cent in june</h
```

Thus for each of these lexical searches, the expression denotes a result set consisting of two match points: the first characters of each matching phrase. Note that the strings "`U.S. `", "`u s u`", and "`..., 'U:'`" all match the phrase "`U.S. up 1.5 per cent in June</h>`":

string	*normalized form*
U.S. up 1.5 per cent ...	u s up 1 5 per cent ...
U.S.	u s
u s u	u s u
..., 'U: '	u

By giving two strings s_1.. s_2 the reader searches for all characters that begin phrases such that the normalized phrase matches either normalized s_1 or normalized s_2 as above or it follows normalized s_1 and precedes normalized s_2 in lexicographic order. In many applications this kind of search capability is very useful and may often replace a long sequence of searches for one string at a time (Logan–Logan 1988). As an example consider the text

```
shortages hit in 1973 and 1979. In the 1980s ... in 1978 ...
```

For the expression "hi".."jo", the result set contains four match points corresponding to the initial characters of the phrases

```
hit in 1973 and 1979. In the 1980s ... in 1978 ...
in 1973 and 1979. In the 1980s ... in 1978 ...
In the 1980s ... in 1978 ...
in 1978 ...
```

which can be represented diagrammatically by the notation

```
shortages hit in 1973 and 1979. In the 1980s ... in 1978 ...
          |   |                 |               |
```

For the expression "1975".."1980" the result set contains three match points:

```
shortages hit in 1973 and 1979. In the 1980s ... in 1978 ...
                            |            |            |
```

4.2. Position search

Position search is used to find a character in a given position in the whole text, or characters at a given distance to the left or right of match points in a match point set. The expression for the first kind of search is

$$[n]$$

where n is a positive integer. As an example, with respect to the following text the PAT expression "[15]" denotes the 15^{th} character, namely the "u" in "Yugoslavs", and thus the indicated match point:

```
Some 700,000 Yugoslavs live in Germany, most of them Croats.
                  |
```

Unlike for lexical searches, the match point denoted by a position search need not be the initial character of an indexed element.

Match points can be shifted by expressions of the form

$$\text{shift}.n\ e$$

where n is a positive or negative integer and e is an expression. The expression e is considered to specify a match point set: if it is a region expression, the match points correspond to the first characters of the regions. For example, with the expression

shift.3 "1800".."2000"

we can find the two indicated match points in the following text:

```
Fascicles of the OED appeared between 1884 and 1928.
                                      |         |
```

Again the match points in the result set need not correspond to starts of phrases.

4.3. Frequency search

PAT includes two groups of frequency expressions, used to find frequently occurring or repeated substrings in a text. The "signif" expressions are used to find the most frequent substrings, consisting of whole indexed elements, and beginning a phrase that matches a given string. The "lrep" expressions are used to find the longest repeated substrings, each consisting of whole indexed elements, beginning with a phrase that matches a given string. The frequency search expressions are always match point expressions.

Consider the frequency expression

signif *e*

where *e* is a match point expression. This operation first normalizes the character strings starting at match points corresponding to *e* and identifies for each one its prefix up to the first delimiter. From the corresponding match points, "signif" returns the subset associated with the most frequently occurring prefix. In the complete works of Shakespeare (OUP 1988), 792 words begin with the string "thro". The command

signif "thro"

returns a match point set with 329 members, corresponding to the initial characters of each occurrence of the word "through" in the text, that being the most frequent word beginning with "thro".

Using the form

signif.*n e*

a user can specify that extended prefixes, including *n* delimiters rather than stopping at the first one, should be compared. For Shakespeare, the following results can be obtained:

expression	*number of matches*	*matching phrase*
signif.2 "thro"	107	`through the`
signif.3 "thro"	8	`through the world`
signif.4 "thro"	2	`throat our height can`

Using the form

signif.-*n s*

the user can retrieve result sets corresponding to the *n* most frequent normalized prefixes beginning with string *s*. For Shakespeare, the following example illustrates this form:

signif.-10 "thro"

329 matches, text=through
116 matches, text=throw
107 matches, text=through the
77 matches, text=throne
58 matches, text=throat
46 matches, text=throws
35 matches, text=thrown
31 matches, text=throats
21 matches, text=throwing
20 matches, text=throng

Note that since the two word prefix "through the" occurs more frequently than do remaining single words, the result set corresponding to this extension of "through" is returned before, for example, the match points corresponding to "throne". Because a *sequence* of match point sets are returned, this form of the command cannot be used within other PAT expressions.

The operation

lrep *e*

chooses from the match points identified by *e* those having the longest normalized extensions such that there are at least two occurrences of the same extension. The result set consists of the subset of match points corresponding to the first characters of these phrases. Using the form

lrep.*n e*

all repeated phrases having at least *n* characters in the common prefix can be identified. Looking again at Shakespeare,

lrep "thro"

returns a result set containing two match points corresponding to a repeated phrase having 162 characters (including line numbers from the play):

```
                    ..throw incense. Have I caught thee?
22  He that parts us shall bring a brand from heaven
23  And fire us hence like foxes. Wipe thine eyes.
24  The goodyear shall devour ['em, flesh and fell,]

                    ..throw incense. Have I caught thee?
22  He that parts us shall bring a brand from heaven
23  And fire us hence like foxes. Wipe thine eyes.
24  The goodyear shall devour [them, flesh and fell,]
```

which represents a variant edition included in the text. Similarly,

lrep.100 "thro"

returns a result set with four match points, two of which are as above and the other two representing another variant edition:

```
                    ..through itself to that full issue
4   For which I razed my likeness. Now, banished Kent,
5   If thou canst serve where thou dost stand condemned,
6   [So may it come thy master, whom thou lov'st,]

                    ..through itself to that full issue
4   For which I razed my likeness. Now, banished Kent,
5   If thou canst serve where thou dost stand condemned,
6   [Thy master, whom thou lov'st ...]
```

Frequency expressions have been included in PAT for the needs of linguists and editors. In an experiment with users the frequency search operations were found among the most difficult to use (Raymond–Fawcett 1990). Therefore, improvements resulting in more useful frequency search operations are currently being investigated.

4.4. Region definitions

A region definition specifies a region set consisting of new regions. It has the form

docs e_1.. e_2

where e_1 and e_2 are match point expressions. Expression e_1 gives the condition for the first character of a new region and e_2 a condition for the last character. If e_1 or e_2 is a region expression, it denotes the set of the first characters of the argument regions.

Suppose we define

year = docs ("1800 " .."2000 ").. (shift.3 "1800 .."2000 ")

Then both years in our earlier sample text would be regions in the set named by the expression *year:

```
Fascicles of the OED appeared between 1884 and 1928.
                                      ----     ----
```

Alternatively, if a region set named "year" had been defined by the text installer, it could be referenced by the expression "docs year".

From a tagged text we can define regions by matching the tags. For example, if the text consists of articles with headlines, publication dates, authors, and paragraphs, such that each of these parts is denoted by tags, we can define the structure in terms of PAT expressions. Specifically, assuming that headlines are denoted by the tags "<h>" and "</h>", the definition

headline = docs "<h>"..(shift.3 "</h>")

defines a region set called *headline in which each element spans the substring from the first character of the opening tag to the last character of the closing tag.

Each region definition creates a region set independently of other region definitions. There are no constraints on how regions in one set overlap earlier defined regions. However, the regions defined in one definition are not self-overlapping. Thus, if a text includes the following fragment:

```
<h>Editor denies <h>Massive Failure</h> misleading</h> ...
```

the definition of headline given above would include the region corresponding to the substring "<h>Massive Failure</h>" but not the surrounding headline.

4.5. Restriction

Users often require means to select subsets from a given set. PAT provides several binary operators of the form

$$e_1 \; op \; e_2$$

which designate result sets that are subsets of the set corresponding to e_1, consisting of elements that satisfy the condition expressed by "$op\ e_2$". Thus if e_1 is a region expression, the result is a region set; if e_1 is a match point expression, the result is a match point set.

The first operator "including" allows users to find those regions that contain at least one member of a given match point set. For example, consider again the complete works of Shakespeare and assume that the name *speech has been defined to designate the set of regions corresponding to all speeches from all the plays. Hence, the expression

*speech including "wherefore art"

returns the subset of speeches containing match points corresponding to the lexical search, namely, the two speeches:

```
..<S JULIET> <T asd> {(not knowing Romeo hears her)}
                                    <T verse> O Romeo,    +
75  Romeo, wherefore art thou Romeo?
76  Deny thy father and refuse thy name,
77  Or if thou wilt not, be but sworn my love,
78  And I'll no longer be a Capulet.
```

and

```
..<S FLAVIUS>
          <T verse> But wherefore art not in thy shop today?
28  Why dost thou lead these men about the streets?
```

In general, the region expression

$$e_1 \text{ including.}n \; e_2$$

yields a result set consisting of those regions in the set specified by e_1 which contain at least n match points specified in e_2 (with ". 1" as the default). Thus

*speech including.7 "Romeo"

returns one speech (Juliet's final monologue).

Similarly,

e_1 not including.n e_2

returns the complementary subset. Thus, to find all Shakespeare's speeches that include the word "dream" but no word beginning with "sleep", a user can write the expression

(*speech including "dream ") not including "sleep"

If the constraining expression is a region expression, then the match point set used to test membership in the result set consists of the first characters of the regions. Continuing with the previous examples, if *scene designates all scenes from Shakespeare's plays,

*scene including.100 *speech

returns all scenes having 100 or more speeches. However, it is important to recognize that PAT only checks that the *start* of the speech is in the scene, which for nested regions is sufficient for the *whole* speech to be in the scene. Thus it follows that if "line" designates all lines from the plays,

*line including *speech

returns the set of lines that contain starts of speeches, as opposed to those containing complete speeches.

The "including" operator produces a result set that is a subset of a region set. PAT provides other operators that produce a subset of an *arbitrary* set by restricting membership according to the following constraints:

op	*explanation*
^	coincident with some member of e_2
–	not coincident with any member of e_2
fby.n	preceding some member of e_2 by at most n characters
near.n	separated by up to n characters from some member of e_2
within	contained in some region designated by e_2

A user wishing to know which sonnets contain suffixed instances of the word "love" could write the expression

*sonnet including ("lov" – "love ")

assuming prior construction of the set *sonnet. The expression "lov" includes (among many others) match points in the following lines from several sonnets:

```
13  And so of you, beauteous and lovely youth,
 3  Both grace and faults are loved of more and less;
14  Those that can see thou lov'st, and I am blind.
 3  Have put on black, and loving mourners be,
 3  But 'tis my heart that loves what they despise,
13  So true a fool is love that in your will,
 9  Rise, resty muse, my love's sweet face survey
```

the last two of which are also included in the set denoted by the expression "love" and therefore excluded from the difference set.

For all of these operators, the set constraints are based on match points. Thus if a region expression is involved, the constraint on each member is evaluated in terms of the match point corresponding to its first character. For example,

*sonnet fby.100 "love "

returns regions in the set *sonnet for which the word "love" occurs within the first 100 characters; the proximity is measured from the start of the region, not the end. This semantics occasionally causes misunderstandings and errors on the part of some users and should therefore be reconsidered.

In an expression of the form

$$e_1 \text{ within } e_2$$

e_2 must be a region expression. For example,

("lov" – "love ") within *sonnet

returns the match points corresponding to suffixed uses of "love" occurring in sonnets (contrast with the use of "including" above), and

sonnetline = *line within *sonnet

returns the regions corresponding to lines within sonnets. As for the previous operators, when e_1 is a region expression, the constraint is based on the match points corresponding to the starts of the regions, but the result set is a (region) subset of e_1.

4.6. Augmentation

Because PAT provides restriction operations corresponding to set intersections "^" and set difference "–", it also includes the binary operator "+" to indicate set union. As expected, when the two operands are match point expressions, the result set is a match point set including all members of both argument sets. For example, to find all sonnet lines that include the words "boy" or "youth", one can write

BoyOrYouth = *sonnetline including ("boy " + "youth ")

which will return 18 lines. To find in which sonnets such lines appear (assuming prior definition of the region expression *title), a user could write

*title within (*sonnet including *BoyOrYouth)

yielding 16 regions containing the titles. To examine these titles together with the sonnet lines, one could then take the union of the two region sets:

*BoyOrYouth + %

(where % refers to the previous result), yielding the following text:

```
[[Sonnet]] 2
  3 Thy youth's proud livery, so gazed on now,
[[Sonnet]] 7
  6 Resembling strong youth in his middle age,
[[Sonnet]] 11
  4 Thou mayst call thine when thou from youth convertest.
[[Sonnet]] 15
 10 Sets you most rich in youth before my sight,
 12 To change your day of youth to sullied night;
[[Sonnet]] 22
  2 So long as youth and thou are of one date;
[[Sonnet]] 37
  2 To see his active child do deeds of youth,
[[Sonnet]] 41
 10 And chide thy beauty and thy straying youth
[[Sonnet]] 54
 13 And so of you, beauteous and lovely youth,
[[Sonnet]] 60
  9 Time doth transfix the flourish set on youth,
[[Sonnet]] 73
 10 That on the ashes of his youth doth lie
[[Sonnet]] 96
  1 <T verse>Some say thy fault is youth, some wantonness;
  2 Some say thy grace is youth and gentle sport.
[[Sonnet]] 98
  3 Hath put a spirit of youth in everything,
[[Sonnet]] 108
  5 Nothing, sweet boy; but yet like prayers divine
[[Sonnet]] 110
   7 These blenches gave my heart another youth,
[[Sonnet]] 138
  3 That she might think me some untutored youth
[[Sonnet]] 153
 10 The boy for trial needs would touch my breast.
```

It should be noted that the semantics of PAT expressions are different from those of Boolean operators in traditional document retrieval systems. A naïve user wishing to find sonnets that include the word "love" as well as either "boy" or "youth" might write

*sonnet including (("boy " + "youth ") ~ "love")

and be surprised when PAT reports "no match". The semantics of PAT expressions indicate that the correct way to pose this query is

(*sonnet including ("boy " + "youth ")) including "love"

If one of the operands of the union operator is a match point expression and the other is a region expression, a simple union of the sets cannot be performed; PAT instead defines the result to be a match point set, using the match points corresponding to the starts of the argument regions in place of the regions themselves. Similarly, if both arguments are region expressions, but the regions in the corresponding sets overlap, PAT cannot form a simple union, since overlapping regions in one set are disallowed. Thus in this situation, PAT again defines the result to be a match point set, using the match points corresponding to the starts of the regions in place of the regions themselves. As a result,

(*speech including "to be or not to be") + (*line including "dying")

is a *region* expression denoting 58 regions of text, but

(*speech including "to be or not to be") + (*line including "death")

is a *match point* expression denoting 1030 match points in the text, since two lines containing the word "death" occur within the speech. Although these semantics are self-consistent, this feature in PAT is easily misunderstood and therefore changing it should be considered, perhaps by suitably defining the union of overlapping regions (as done, for example, in Burkowski (1991)).

5. Conclusions

We have described the query capabilities included in the PAT text search system. We have divided PAT expressions into six classes and introduced the syntax and semantics of the expressions in the classes. We have shown that PAT indexing can be specified by productions as a view of PAT text seen as a character sequence. The matching of a phrase with a given string was also described by productions and text translation.

During the specification of PAT's search capabilities we have identified some problem areas which are interesting, not only from the point of view of PAT evaluation, but also in the evaluation of search capabilities in text search systems more generally. The indexing definition capabilities in PAT offer a means for application dependent indexing. However, the current indexing definition techniques are limited and designed for homogeneous text, whereas some document collections, such as multilingual collections, might include texts with heterogeneous indexing needs. Open Text Corporation supports a facility known as "Parallel PAT" that provides a single PAT interface to a collection of independently managed PAT texts, each using its own indexing definition.

The flexible definition of regions is an important and original feature in PAT. Together with the indexing definition capability, this feature supports application dependent handling of text. Regions may be defined both during the text installation phase and by the reader. Using PAT expressions the reader may search for either match points or regions, using a uniform syntax and semantics. In most cases the user may consider the restriction operations as if they were truly region operations, but occasionally the semantic implications are surprising. This is even more apparent in the augmentation operation. To support users familiar with more conventional document-based Boolean searching, a variety of front ends should be designed and implemented using PAT as a back-end search engine invoked through the program interface using PAT expressions. To date, experience with a restricted front end that provides simple look-ups in the *Oxford English Dictionary* and with a graphics-based front end to search an automobile engine manual, among others, shows that the separation of end-user convenience from search capabilities can be successful.

PAT with its region definition capabilities is designed for handling structured text. The structure of text is often expressed by tags, which can be used to define most regions. After the region definitions, the user might want to handle the text in the form where all tags are hidden. In the workstation environment, LECTOR allows a user to read regions without reading tags. For example, the reader of the *Oxford English Dictionary*, Shakespeare, or the Bible can read text displayed in a form similar to that in the printed volumes. However, the search is always applied to the tagged text, requiring users to be aware of the tags. Through the introduction of regular expressions in the definition of PAT indexing and search, more convenient search can be supported, wherein, for example, standard markup tags can be defined to be delimiters. The development of text search systems where the user performs *all* operations on text in various forms is an area of ongoing research.

Finally, the unique PAT indexing technique has made it possible to implement frequency search operations. However, the semantics of the current frequency operations is difficult to define, and they are limited in utility. Thus further studies for the development of these operations are needed.

Acknowledgements

Gaston Gonnet was the principal designer and implementer of PAT; several others, including Tim Snider and Byron Weber Becker, contributed substantially to the implementation. Financial support from the Natural Sciences and Engineering Research Council of Canada, under grants CRD0000862 and STR0045480, and from the Academy of Finland is gratefully acknowledged.

References

Berg, D.L.–Gonnet, G.H.–Tompa, F.W. 1991. The New Oxford English Dictionary Project at the University of Waterloo. In: Zampolli, A.–Cignoni, L.–Peters, C. (eds): Computational Lexicology and Lexicography: Special Issue Dedicated to Bernard Quemada. Series Linguistica Computazionale, vol. VII, 29–44. Giardini Editori, Pisa.

Burkowski, F.J. 1991. Textriever: A retrieval engine for multimedia databases. In: Proc. of the Int. Conf. on Multimedia Information Systems. Singapore, 71–6.

Croft, W.B. 1984. Implementing a text storage and retrieval tool for the office. In: Proc. of the First Int. Conf. on Office Automation. IEEE Computer Society, 137–44.

Faloutsos, C. 1985. Signature files: Design and performance comparison of some signature extraction methods. In: Proc. of ACM-SIGMOD 1985, Int. Conf. on Management of Data. SIGMOD Record 14: 63–82.

Goldfarb, C.F. 1990. The SGML Handbook. Oxford University Press.

Gonnet, G.H. 1983. Unstructured databases—or—very efficient text searching. In: Proc. of ACM Principles of Database Systems.

Gonnet, G.H. 1987. Examples of PAT applied to the Oxford English Dictionary. Tech. Rept. OED-87-02, UW Centre for the New OED and Text Research, University of Waterloo.

Gonnet, G.H.–Tompa, F.W. 1987. Mind your grammar: a new approach to modelling text. In: Proc. of the 13th Int. Conf. on Very Large Data Bases, 339–46.

Gonnet, G.H.–Baeza-Yates, R.A.–Snider, T. 1991. Lexicographical indices for text: inverted files vs. PAT trees. Tech. Rept. OED-91-01, UW Centre for the New OED and Text Research, University of Waterloo. Also in: Frakes, W.B.–Baeza-Yates, R. (eds): Information Retrieval: Data Structures and Algorithmus, 66–82. Prentice–Hall, 1992.

Gonnet, G.H.–Snider, T. 1992. Mealy Machines. Unpublished manuscript. UW Centre for the New OED and Text Research, University of Waterloo.

Hollaar, L.A. 1979. Text retrieval computers. In: Computer 12: 40–50.

Larson, P.-A. 1984. A method for speeding up text retrieval. In: ACM Data Base 15/2: 19–23.

Lee, D.L. 1985. The design and evaluation of a text-retrieval machine for large databases. Ph.D. thesis. Computer Systems Research Institute, University of Toronto.

Logan, H.M. – Logan, G. 1988. An inquiry into inquiry systems: a discussion of some applications of data retrieval to the New OED database. In: Proc. of the Fourth Conf. of the UW Centre for the New OED and Text Research. "Information in Text". Waterloo, ON, 26–28 October, 1988, 81–95.

Open Text Corporation. 1989–93. PAT Reference Manual and Tutorial.

Open Text Corporation. 1993. PATMOTIF Tutorial.

Oxford University Press. 1988. Complete Electronic Shakespeare.

Raymond, D.R. – Fawcett, H.J. 1990. Playing detective with full text searching software. In: Proc. of SIGDOC '90, SIGDOC Asterisk 14: 157–66.

Raymond, D.R. 1992. Flexible Text Display with LECTOR. In: Computer 25/8.

Salminen, A. – Tompa, F.W. 1992. Data modelling with grammars. Unpublished manuscript. UW Centre for the New OED and Text Research, University of Waterloo.

Salton, G. 1981. A blueprint for automatic indexing. In: ACM SIGIR Forum 16/2: 22–38.

Salton, G. – McGill, M.J. 1983. Introduction to Modern Information Retrieval. McGraw-Hill, New York.

Salton, G. 1989. Automatic Text Processing. Addison-Wesley, Reading.

Tague, J. – Salminen, A. – McClellan, C. 1991. A complete model for information retrieval systems. In: Proc. of the 14th Int. ACM/SIGIR Conf. on Research and Development in Information Retrieval, 14–20.

Tompa, F.W. 1992. An Overview of Waterloo's Database Software for the OED. Proc. Symp. on Historical Dictionary Databases and Data Retrieval Requirements (edited by T.R. Wooldrich) (Toronto, October 1991). In: Centre for Computing in the Humanities, Working Papers 2: 123–143.

Appendix

The syntax of PAT expressions

```
PAT_expression ::=
    [match_point_name '='] match_point_expr |
    [region_name '='] region_expr |
    match_point_expr_sequence.

match_point_expr ::=
    lexical_search |
    position_search |
    frequency_search |
```

```
    match_point_restriction |
    match_point_augmentation |
    '*' match_point_set_name |
    match_point_set_number |
    '%' |
    '(' match_point_expr ')' |
    region_expr.

lexical_search ::=
    string |
    string '..' string.

position_search ::=
    '[' positive_integer ']' |
    'shift' '.' integer match_point_expr.

frequency_search ::=
    'signif' [ ['.' positive_integer ] match_point_expr ] |
    'lrep' [ ['.' positive_integer ] match_point_expr ].

match_point_expr_sequence ::=
    'signif' '.' negative_integer string.

match_point_restriction ::=
    match_point_expr '^' match_point_expr |
    match_point_expr '-' match_point_expr |
    match_point_expr [ 'not' ] 'fby' [ '.' positive_integer ] match_point_expr |
    match_point_expr [ 'not' ] 'near' [ '.' positive_integer ] match_point_expr |
    match_point_expr 'within' region_expr.

match_point_augmentation ::=
    match_point_expr '+' match_point_expr.

region_expr ::=
    region_definition |
    'docs' installed_region_definition |
    region_restriction |
    region_augmentation |
    '*' region_set_name |
    region_set_number |
```

```
    '%' |
    '(' region_expr ')'.

region_definition ::=
    'docs' match_point_expr '..' match_point_expr.

region_restriction ::=
    region_expr [ 'not' ] 'including' [ '.' positive_integer ] match_point_expr |
    region_expr '^' match_point_expr |
    region_expr '-' match_point_expr |
    region_expr [ 'not' ] 'fby' [ '.' positive_integer ] match_point_expr |
    region_expr [ 'not' ] 'near' [ '.' positive_integer ] match_point_expr |
    region_expr [ 'not' ] 'within' region_expr.

region_augmentation ::=
    region_expr '+' region_expr.
```

N.B. An expression of the form

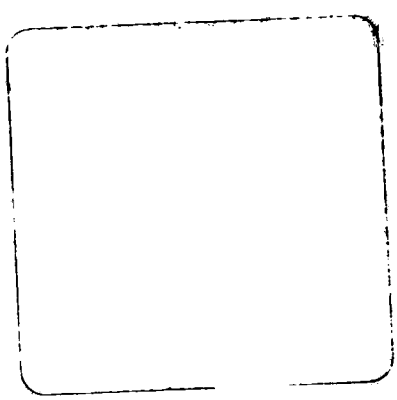
region_expr '+' region_expr

is a match point expression if some region specified by the first region expression overlaps some region specified by the other region expression.

Addresses of the authors: Airi Salminen
Department of Computer Science
University of Jyväskylä
Seminaarinkatu 15
40100 Jyväskylä
Finland

Frank W. Tompa
Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada

PRINTED IN HUNGARY
AKADÉMIAI KIADÓ ÉS NYOMDA VÁLLALAT
BUDAPEST



Authors are requested to send two hard copies of their manuscript + a floppy disk with the following specifications:

operation system: DOS 2.0 or later;
floppy disk size: 5.25 or 3.5 inch;
floppy disk format: DS,DD (360/720 Kbyte) or DS,HD (1.2/1.44 Mbyte);
text editor: XYWrite, Word or Word Perfect;
text file: ASCII file without the formatting commands of the text editor;
special characters: choose some character combinations and use them consequently, for example: ə=\schwa{}; ɔ=\openo{}.

Manuscript should be accompanied by an abstract of about 100 words. It should be typed on a separate sheet of paper.

Tables, diagrams and illustrations (with the author's name and an Arabic number) should be presented on separate sheets. Captions should be typed on a separate sheet and placed at the end of the manuscript.

Footnotes should be typed double-spaced on a separate sheet at the end of the article. All language examples (and only these) are to be italicized (single underlining).

Citations in the text should be enclosed in double quotation marks (" " in case of a paper written in English, „ " in German and « » in French).

Reference to a publication should be made by the name of the author, the year of publication and, when necessary, by page numbers in the following ways:

. . . as described by Schmidt (1967) . . .
. . . as referred to by Hoover (1967, 56–78; 1976, 43).
. . . mentioned by several authors (Elgar 1978, 77; Williams 1981, 154–6) . . .

An alphabetically arranged list of references should be presented at the end of the article as follows:

Bárczi, G. 1958a. Magyar hangtörténet [The history of Hungarian sounds]. Akadémiai Kiadó, Budapest.

Bárczi, G. 1958b. A szótövek [Word stems]. Akadémiai Kiadó, Budapest.

Lakoff, G.–Peters, P.S. 1969. Phrasal conjunction and symmetric predicates. In: Reibel, D.–Schane, S. (eds): Modern Studies in English, 113–41. Prentice–Hall, Englewood Cliffs.

Leben, W.R. 1980. A metrical analysis of length. In: Linguistic Inquiry 11: 497–509.

Ross, J.R. 1967. Constraints on Variables in Syntax. Ph.D. dissertation. MIT, Cambridge MA.

For marking subsections decimal notation should be applied. Do not use more than four digits if possible.

Examples within the text should be marked in italics. Meanings are to be rendered between inverted commas (' '). If glosses are given morpheme by morpheme, the initial letter of the gloss should be placed exactly below that of the example. Grammatical morphemes can be abbreviated in small case letters connected to the stem or the other morphemes by a hyphen. No period should be applied in such abbreviation. For example:

(1) (a) A sólymaid elszálltak
the falcon-gen-pl-2sg away-flew-3pl
'Your falcons have flown away.'

Examples can be referred to in the text as (1a), (1a–d), etc.

One proof will be sent to the author. Please read it carefully and return it by air mail to the editor within one week: Acta Linguistica Hungarica, MTA Nyelvtudományi Intézet, H-1250 Budapest, P.O. Box 19.